ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación:

INGENIERO TÉCNICO EN INFORMÁTICA DE GESTIÓN

Título del proyecto:

"DESARROLLO DE UN SISTEMA SCADA BASADO EN LABVIEW PARA LA GESTIÓN DE UNA MICRORRED ENERGÉTICA"

Alumno: María Muñoz Santos

Tutor: Jesús María Corres Sanz

Pamplona, 30 de Junio de 2011

ÍNDICE:

1. Introducción 3
1.1. Resumen Proyecto 3
1.2 Aportación personal 4
1.3. Sistema SCADA 5
1.4. Herramientas utilizadas 5
1.4.1 LabVIEW 5
1.4.2. PXi 6
1.4.3. PC 6
1.4.4 SGBD 6
2. Análisis de la aplicación 7
2.1. Origen de los datos 7
2.2. Estructura 8
2.3. Requisitos del proyecto 8
2.3.1 Requisitos software 9
2.3.2 Requisitos hardware 10
2.3.3 Requisitos de usuario 10
2.4. Análisis del sistema 11
2.4.1. Diagramas de flujo de datos 11
3. Diseño 14
3.1. Diseño Base de Datos 14
3.1.1. Información general 15
3.1.1.1 SGBD 15
3.1.1.2. SQL Server 16
3.1.1.3. ODBC 17
3.1.2. Creación de la Base de Datos 18
3.1.2.1. Modelo Entidad – Relación 21
3.1.2.2. Modelo Relacional 24
3.1.3. Mantenimiento de la Base de Datos 40
3.1.3.1. Microrred Maintenance Plan 42
3.1.3.2. Differential Back Up 44
3.1.3.3. Log Transaction Back Up 45
3.2. Diseño LabVIEW 46

3.2.1. Explicación general 46
3.2.2. Esquema General 47
3.2.3. Esquema General Térmico 54
3.2.4. Esquema de Comunicación 56
3.2.5. Gráficas en tiempo real 62
3.2.6. Históricos 71
3.2.7. Selección de datos 85
3.2.8. FTP 90
3.2.9. Interacción con Base de Datos 96
3.2.10. Librerías creadas 102
a) Global 104
b) SQLToolsLLB 105
c) ArrayToolsLLB 115
d) DBToolsLLB 123
e) FileToolsLLB 140
f) ChartToolsLLB 145
g) StringAndNumericTools 149
4. Manual de usuario 156
4.1. Introducción 156
4.2. Arrancar y parar sistema 156
4.2.1. Arrancar sistema 156
4.2.2. Parar el sistema 157
4.3. Cambios en la Microrred 158
4.3.1. Añadir módulo 158
4.3.2. Eliminar módulo 167
4.3.3. Añadir atributo a módulo 173
4.3.4. Eliminar atributo de módulo 179
4.4. Cambios en la Base de Datos 184
4.4.1. Información Base de Datos 184
4.4.2. Crear Base de Datos 185
4.4.3. Eliminar datos 186
4.4.4. Eliminar Base de Datos 187
4.4.5. Restaurar Base De Datos 188

1. Introducción

ÍNDICE:

1.1. Resumen Proyecto
1.2. Aportación personal
1.3. SCADA
1.4. Herramientas utilizadas
1.4.1. LabVIEW
1.4.2. PXi
1.4.3. PC
1.4.4. SGBD

1.1. Resumen Proyecto

Este proyecto se enmarca dentro de un proyecto mayor basado en el desarrollo de una microrred eléctrica en el departamento de Ingeniería Eléctrica y Electrónica de la UPNA. Ese proyecto tiene como finalidad diseñar microrredes eléctricas en las que se implementen estrategias de control para lograr la optimización de sus elementos añadiéndoles nuevas funcionalidades, garantizando el suministro eléctrico de las cargas en funcionamiento aislado, atenuando las perturbaciones introducidas en el PCC en funcionamiento conectado a red, y colaborando con la red eléctrica en el mantenimiento de su estabilidad. Asimismo será una característica muy importante su capacidad de modularidad, es decir, de permitir la conexión de nuevos elementos según el concepto plug-and-play.

La microrred está ubicada en el Campus Arrosadía (Pamplona) de la Universidad Pública de Navarra tiene una potencia estimada de generación eléctrica máxima de 20kW, teniendo en cuenta los perfiles combinados de generación eólica y fotovoltaica, y el apoyo del grupo electrógeno y pila de combustible. Lógicamente, la potencia total instalada en los elementos será mayor, y su valor se incluye dentro de la descripción de los mismos, a continuación.

La microrred está compuesta por los siguientes elementos:

- Aerogenerador de 6kW.
- Generador fotovoltaico de 5kW formado por 60 módulos BP585.
- Grupo electrógeno de 16,5 kVA.
- Sistema de hidrógeno, incluyendo sistema de electrolisis de potencia estimada 5-10kW, sistema de almacenamiento de 35Nm^3 y sistema de pila de combustible, de 5-10KW.
- Banco de baterías, de capacidad máxima de almacenamiento estimada de 100kWh.

- Sistema de acondicionamiento eléctrico (electrónica de potencia) basada en arquitectura mixta.
- Sistema de comunicaciones
- Sistema de monitorización
- Cargas trifásicas controlables para ensayos experimentales de la microrred.
- Interruptor controlado en la conexión a la red eléctrica con objeto de hacer funcionar a la microrred tanto en modo aislado como conectado a red.

El objetivo principal de la microrred es el de constituir una instalación singular para la investigación y desarrollo tecnológico de microrredes eléctricas. Por ello, en su configuración y diseño se ha tenido en cuenta la posible integración futura de nuevos dispositivos, la modularidad necesaria para integrar equipos adicionales que eleven la potencia del sistema o gestionen de distinta manera la energía, y el uso de la microrred como banco de ensayos de nuevos sistemas de acondicionamiento eléctrico, comunicaciones y monitorización.

La microrred establecida cuenta con diversos dispositivos de almacenamiento y generación eléctrica, todos ellos relacionados con las energías renovables. Así pues, un dispositivo centra, el inversor híbrido, está conectado a diversos dispositivos. Como fin último se desean desarrollar unas estrategias eléctricas que permitan optimizar la utilización de energías renovables.

Su funcionamiento básico es el siguiente:

El inversor híbrido recoge datos de distintas variables de todos los elementos de la red. Fundamentalmente, según los valores de las potencias de los distintos dispositivos decide si debe inyectarse corriente a la red o almacenarla en dispositivos de almacenamiento como las baterías.

1.2 Aportación personal

Todos los dispositivos de la microrred previamente comentada generan una serie de datos que resultan interesantes de almacenar tanto para la visualización en tiempo real del estado de la microrred como para la generación de informes sobre un tiempo determinado.

La generación de los datos es constante y su almacenamiento importante, así que como estudiante de Ingeniería Técnica en Informática de Gestión, se me encomienda la tarea del almacenamiento, gestión y presentación de dichos datos. El software a realizar debe ser fiable, sin pérdida de datos y consistente, así como fácil de utilizar y entender para los posteriores trabajadores en el mismo.

Este documento trata de relatar todas las fases que se han producido en el desarrollo de este software, desde el establecimiento de requisitos, el diseño y la implementación hasta la puesta en funcionamiento del mismo.

1.3. Sistema SCADA

SCADA proviene de Supervisory Control And Data Acquisition, o Control Supervisor y Adquisición de Datos. Es pues, un sistema que monitoriza y controlar un lugar en concreto.

En un sistema SCADA existe un ordenador que realiza automáticamente las tareas de supervisión, tratamiento de datos y control de procesos. Se ejecuta en tiempo real, y un operador puede interactuar con el sistema para supervisar y controlar los procesos. Además, será el sistema el que provea al usuario toda la información requerida.

Tantos los programas como el software adicional se denominan en general sistema SCADA.

El sistema SCADA se ejecuta automáticamente, pero un usuario puede interactuar con el sistema con el fin de realizar modificaciones, y será el sistema el que provea toda la información generada al usuario.

Es por eso que el proyecto que describe este documento es considerado un sistema SCADA, ya que esencialmente se trata de una serie de dispositivos que interactúan con un ordenador, generan y se comunican datos, y todo ello es tratado para ofrecer al usuario los resultados oportunos.

1.4.Herramientas utilizadas

1.4.1 LabVIEW

Uno de los requisitos del proyecto era realizarlo íntegramente en LabVIEW, tanto la comunicación de datos como la interacción con el usuario.

LabVIEW es una herramienta gráfica de programación utilizada para pruebas, control y diseño. Utiliza un lenguaje G, es decir, un lenguaje gráfico, que es un lenguaje de programación de cuarta generación. Fue desarrollado por National Instruments en 1976, aunque actualmente la última versión disponible es la de 2010. Sus programas son llamados Instrumentos virtuales o VIs, aunque actualmente las capacidades de los mismos se extienden mucho más allá que el control de instrumentos. También es posible crear subprogramas o subVIs, que aportarán mayor funcionalidad al VI principal. LabVIEW es utilizado a nivel mundial por miles de científicos e ingenieros en sistemas de control de instrumentos, adquisición de datos, automatización industrial, etc.

Sus principales características son una programación rápida y sencilla, debido al lenguaje gráfico que utiliza. Así pues, no hace falta ser un experto en programación para desarrollar programas, ya que se trata de una herramienta muy intuitiva a la par que potente. Además, es capaz de interactuar con otros lenguajes y con otras aplicaciones,

como por ejemplo un sistema gestor de bases de datos (SGBD). Es pues, una herramienta muy acertada para desarrollar el software que nos ocupa, ya que presenta unas características muy adecuadas para el manejo tanto de datos como de instrumentación. Se ha utilizado, pues, el siguiente software:

-LabVIEW 2010 Full Development System
-LabVIEW 2010 Database Connectivity Toolkit

1.4.2. PXi

Como complemento a LabVIEW, una parte del sistema se ejecutará sobre un PXi. Un PXi es un ordenador específicamente desarrollado por National Instruments para trabajar con LabVIEW. Su característica principal es la posibilidad de conectar hardware de distinta naturaleza y recoger y almacenar los datos de manera centralizada.

1.4.3. PC

PC de propósito general donde se ejecutarán los programas con los que interactuará el usuario. Además, alojará todos los datos almacenados. Tiene instalado un sistema operativo Windows 7 de 64 bits.

1.4.4 SGBD

SGBD proviene de database management system, o sistema gestor de bases de datos. Los SGBD son un software muy específico, cuya tarea es establecer una conexión entre el usuario y la base de datos, así como las aplicaciones utilizadas. Además, realiza una serie de tareas de mantenimiento sobre las bases de datos, para mantenerlas operativas y que funcionen eficientemente.

Las principales ventajas de un SGBD son la independencia de los datos y el acceso eficiente y concurrente a los mismos. Además, asegura la integridad de los datos de modo que sean los correctos. Además, proporciona unas herramientas de seguridad interesantes de ser consideradas.

Es por todo ello que resulta imprescindible dotar al proyecto de un SGBD adecuado, de manera que los datos estén perfectamente organizados y sean totalmente accesibles. Un SGBD es muy superior en todos los aspectos a otros sistemas de almacenamiento, como pueden ser Microsoft Excel o Microsoft Access. Con un SGBD te aseguras la perdurabilidad y la persistencia de los datos en el tiempo. Además, es mucho más potente en el manejo de los mismos, haciendo las búsquedas más eficientes y rápidas.

Por todo ello, se ha decidido incluir en el proyecto un SGBD, eligiendo posteriormente entre los disponibles el más adecuado de acuerdo a las características y los requisitos del proyecto.

2. Análisis de la aplicación

ÍNDICE:

2. Análisis de la aplicación
   2.1. Origen de los datos
   2.2. Estructura
   2.3. Requisitos del proyecto
      2.3.1. Requisitos software
      2.3.2. Requisitos hardware
      2.3.3. Requisitos de usuario
   2.4. Análisis del sistema
      2.4.1. Diagramas de flujo

2.1. Origen de los datos

La microrred se encuentra en el laboratorio de energías renovables (Lab EERR), dentro del edificio de los pinos. Está estructurada de modo que un dispositivo central se comunique con diversos dispositivos. Los dispositivos involucrados son los siguientes:

Inversor híbrido: Núcleo de la microrred, que está conectado a todos los dispositivos, tanto los propiamente relacionados con las energías renovables como los dos ordenadores. Se divide en tres partes: Híbrido cargador, Híbrido Inversor e Híbrido Eólica. Se conecta al PXi mediante RS-485(9.600bps).

Aerogenerador: ubicado en el campus de la universidad, en la parte trasera del edificio de los pinos. Se conecta a la parte de híbrido eólica del inversor y al PXi mediante Ethernet.

Célula fotovoltaica: ubicadas sobre el techo del edificio de los Pinos. Se conecta a la parte de híbrido cargador del inversor y al PXi mediante Ethernet.

Hidrógeno: ubicado en el Lab EERR. Se conecta a la parte de híbrido cargador del inversor, mediante Ethernet y RS-485(19.200bpx) al PXi.

Baterías: ubicadas en el Lab EERR. Se conectan a la parte de híbrido cargador del inversor y mediante RS-485 al Pxi.

Ultracondensadores: ubicados en el Lab EERR. Se conectan a la parte de híbrido cargador del inversor y mediante RS-485(9.600bpx) al Pxi.

Cargas críticas: Se conectan a la parte del híbrido inversor del inversor y mediante Ethernet al PXi.

Cargas no críticas: Se conectan a la parte del híbrido inversor del inversor y mediante RS-485(19.200bps) al PXi.

Grupo Diesel: Se conecta a la parte del híbrido inversor del inversor y mediante RS-485(19.200bps) al PXi.

RED: Se conecta a la parte del híbrido inversor del inversor y mediante RS-485(19.200bps) al PXi.

Meteorológicas: Se conectan al PXi mediante 4-20mA (analógico).

Pxi: Recoge todos los datos generados en el sistema por diferentes medios comentados en el texto anterior.

PC de propósito general: es el destino final de todos los datos generados en el sistema, donde se gestionarán, almacenarán y tratarán adecuadamente. Está conectado usando un cable Ethernet al Pxi, desde el cual le llegan los datos. También puede acceder a ficheros concretos del Pxi mediante el protocolo FTP.

2.2. Estructura

La estructura de la aplicación es totalmente centralizada. Mientras que el PXi se encarga de recoger todos los datos generados en la microrred con una frecuencia determinada, 1 segundo, es en el PC donde se ejecutan todos los programas que devuelven datos interesantes para el usuario. Una vez desarrollado el sistema, no se requieren varios trabajadores que se encarguen del mismo, si no que basta con un solo supervisor, por lo que no se produce comunicación entre diversas personas, ya que una sola se encarga del sistema.

El sistema está pensado para generar una gran cantidad de datos, pero son unos datos específicos y totalmente acotados, que si bien cambian en valor, no así lo hacen en cantidad. Es por eso que aunque se generen muchos datos, el hecho de que estén perfectamente definidos facilitará enormemente la tarea de gestión y almacenamiento de los mismos.

Así pues, resumiendo, el sistema se ejecutará localmente en un ordenador de propósito general, que también almacenará los datos generados por todo el complejo de la microrred, que a su vez serán recogidos por el PXi, y transmitidos al PC.

2.3. Requisitos del proyecto

Antes de comenzar cualquier tipo de desarrollo del sistema, había que hablar con los desarrolladores y usuarios del sistema, para poder comprender totalmente las necesidades y los requerimientos de la aplicación a realizar.

Habría que saber separar de todo lo comentado lo realmente imprescindible para la aplicación y lo que resultaría opcional o “añadido”, de modo que se diese prioridad a las actividades pertenecientes al primer grupo por encima de las segundas.

Es cierto que a medida que se produce el desarrollo de la aplicación se producirán cambios en las funcionalidades, pero los requisitos básicos continuarán siendo los mismos, por eso es importante esclarecerlos totalmente.

2.3.1 Requisitos software

Básicamente, se buscaba un software centralizado, que sería usado por un grupo reducido de personas e implementado en LabVIEW.

El software debía ser muy seguro y muy fiable respecto al almacenamiento de los datos, así que era necesario asegurarse una pérdida mínima de los mismos.

Un dato muy importante es la posibilidad de ampliación del software, ya que era muy probable la ampliación de la microrred con algún otro dispositivo, la modificación en al uno ya existente o la eliminación de alguno. Es por eso que la aplicación debía ser totalmente flexible, por lo que un diseño modular, en el que cada dispositivo se tratase independientemente, sería lo más adecuado.

La aplicación a realizar se podría diferenciar en tres partes:

- Información en tiempo real del estado de la microrred.

En este apartado es fundamental la representación correcta del estado de la microrred, ya que ayudaría a detectar errores en la misma, de modo que unos datos incorrectos llevarían a una conclusión incorrecta. Además, se ejecutaría en tiempo real, por lo que los posibles errores pudiesen ser detectados de inmediato, y la información representada fuese veraz.

- Gráficas en tiempo real de los datos generados por la microrred.

Así como en la información en tiempo real, estas gráficas contendrían información totalmente actualizada del sistema. Estos datos deben ser correctamente comprobados ya que se usarán como medio de supervisión del sistema.

- Históricos que permitieran ver datos generados previamente

Se genera una gran cantidad de datos que debe ser correctamente almacenada y tratada. Estos datos deberán poderse visualizar mediante una interfaz amigable en forma de gráfica, que realice las operaciones oportunas con los datos requeridos.

2.3.2 Requisitos hardware

Los datos deberían poderse almacenar al completo, sin producirse la pérdida de los mismos, por lo que habría que asegurar un espacio suficiente para ellos.

La manipulación de los datos es constante, ya que el sistema estaba pensado de modo que se produjesen datos ininterrumpidamente cada segundo, y por lo tanto, una vez por segundo, se produjese una acción que almacenase los datos. Es por eso que el sistema debe ser desarrollado en tiempo real, y debe ser rápido en ejecución para no producir retrasos ni diferencias entre los datos generados y los mostrados por pantalla.

Debido a la importancia de los datos, habría que generar un sistema de copias de seguridad, de modo que si se produjese algún imprevisto, se podrían recuperar todos los datos almacenados previamente.

2.3.3 Requisitos de usuario

Por parte de los usuarios, también se establecieron una serie de requisitos.

Probablemente, los futuros usuarios del sistema, serían titulados en Ingeniería Industrial, y no en alguna rama de la informática. Es por eso que, aunque serán capaces de manejar la información que les provea el sistema muy eficientemente, debido a todos los conocimientos sobre ello, no pasará lo mismo con la base de datos que se instalará. Se presupone un no conocimiento por parte de los futuros usuarios para gestionar una base de datos del tamaño y la complejidad estimados.

Por esto, se elaborarán herramientas de apoyo a la hora de trabajar con la base de datos. En caso de querer añadir o quitar un dispositivo, correspondería añadir o quitarlo en la base de datos, sin embargo, hacer eso sin el suficiente conocimiento puede tener unas consecuencias catastróficas para la base de datos. Así pues, se implementarán en LabVIEW una serie de programas para añadir, modificar, eliminar o consultar los dispositivos en la base de datos. Estas interfaces realizarán todas las conexiones oportunas con el resto de la base de datos, para integrarse adecuadamente.

Además, todo deberá estar perfectamente documentado en un manual de uso, de forma que los pasos para actualizar el sistema en caso de querer ampliarlo, etc, estén correctamente definidos.

2.4.Análisis del sistema

El sistema puede ser analizado a alto nivel de modo que sea posible apreciar sus principales funcionalidades. Por ello, se han desarrollado los diagramas de flujo correspondientes.

2.4.1. Diagramas de flujo de datos

Los diagramas de flujo de datos (DFD) son un tipo de herramienta de modelado de sistemas que representa el flujo de información y las transformaciones aplicadas a los datos que se mueven desde la entrada a la salida. La notación procede de la teoría de grafos. Los DFD se van refinando para ampliar la información y aportar un mayor detalle funcional.

En la Figura 2.1 se puede ver el diagrama de contexto o nivel 0, que representa el sistema y las entidades externas con las que se relaciona.

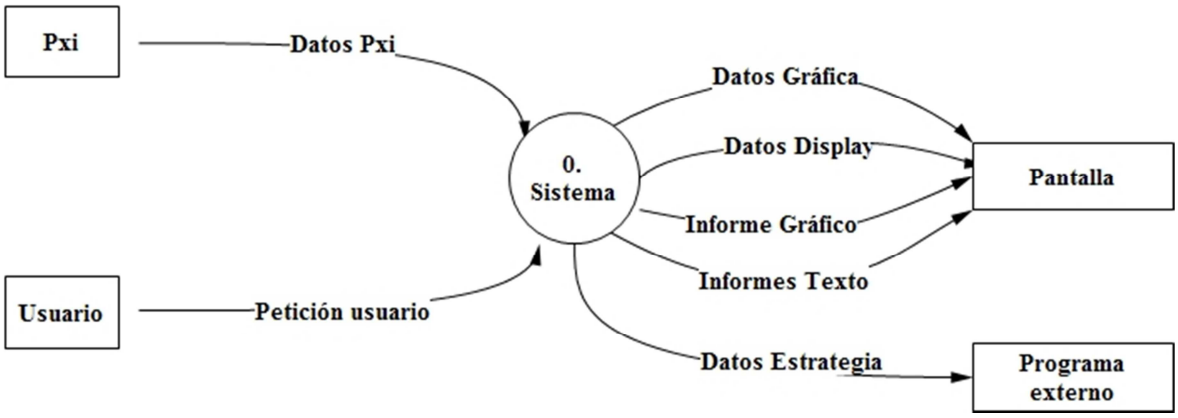

Figura 2.1: Diagrama de contexto del sistema

Se puede ver que el sistema recibe datos desde el Pxi y también peticiones por parte del usuario. Después del procesamiento oportuno devolverá a la pantalla datos que serán representados en una gráfica o en un display. También puede devolver informes, ya sean gráficos o de texto. Además, existe la opción de que devuelva datos para ser ejecutados en un programa externo. La función de este programa externo es ejecutar distintas estrategias sobre un mismo grupo de datos que se han generado en las mismas condiciones meteorológicas. De este modo, se hace posible comparar distintas estrategias y comprobar su eficiencia, así como cuál obtiene mayor rendimiento de la energía generada por las fuentes de energía renovable.

En la Figura 2.2 podemos ver el proceso 0 refinado, es decir, el nivel 1 de refinamiento.

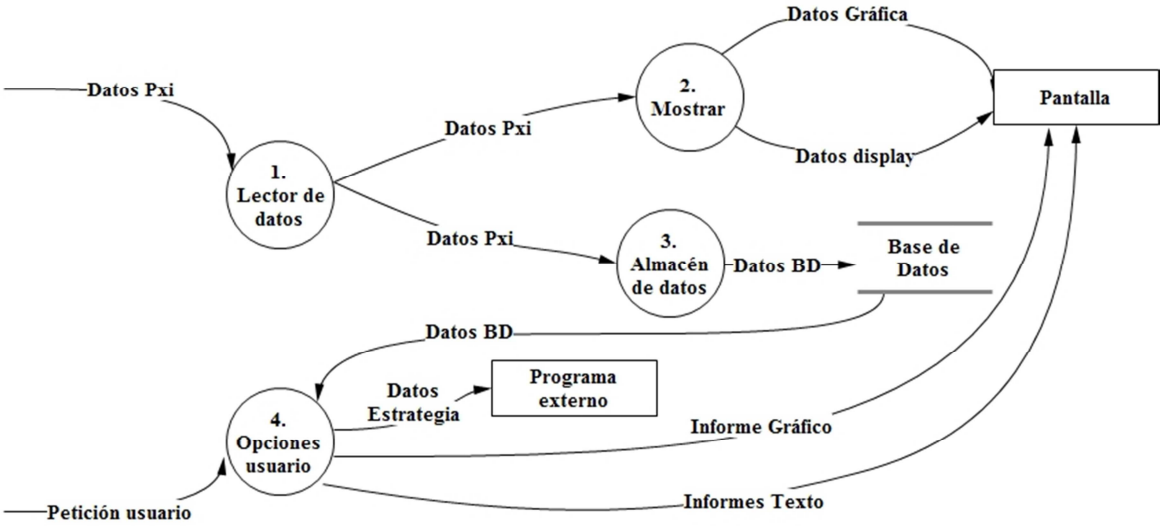

Figura 2.2: DFD de nivel 1 del sistema

En este diagrama se pueden ver las funciones principales del sistema. Por un lado, los datos provenientes del Pxi son leídos y se mostrarán por pantalla, tanto en forma de gráfica como en displays numéricos. Además, estos datos serán tratados para almacenarse en una base de datos.

Por otro lado tenemos peticiones por parte del usuario. Según la petición y los datos obtenidos de la base de datos, se obtendrán datos que serán ejecutados en un programa externo. Además, el usuario podrá pedir informes tanto en una gráfica como textuales.

Se puede refinar algo más el proceso 2.Mostrar, como se ve en la Figura 2.3.

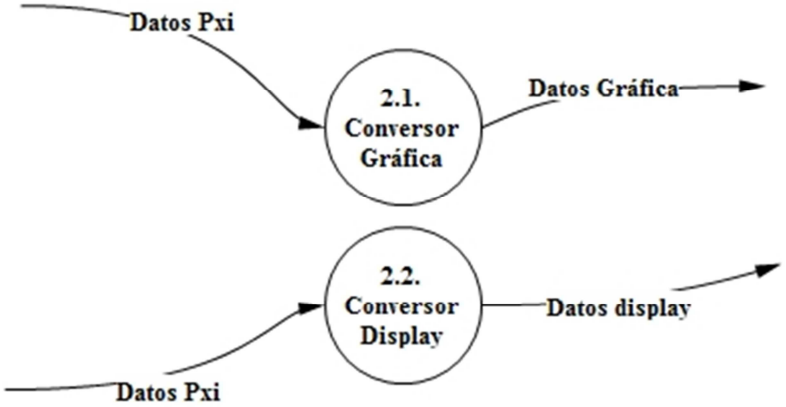

Figura 2.3: DFD de nivel 2 del proceso “Mostrar”

En este diagrama se puede apreciar que, aunque se partan de los mismos datos (Datos Pxi), que son los correspondientes a todos los dispositivos de la Microrred, se tratan de modo diferente según se quieran representar en una gráfica o en un display.

También se ha considerado interesante refinar el proceso 4. Opciones usuario, como se ve en la Figura 2.4.

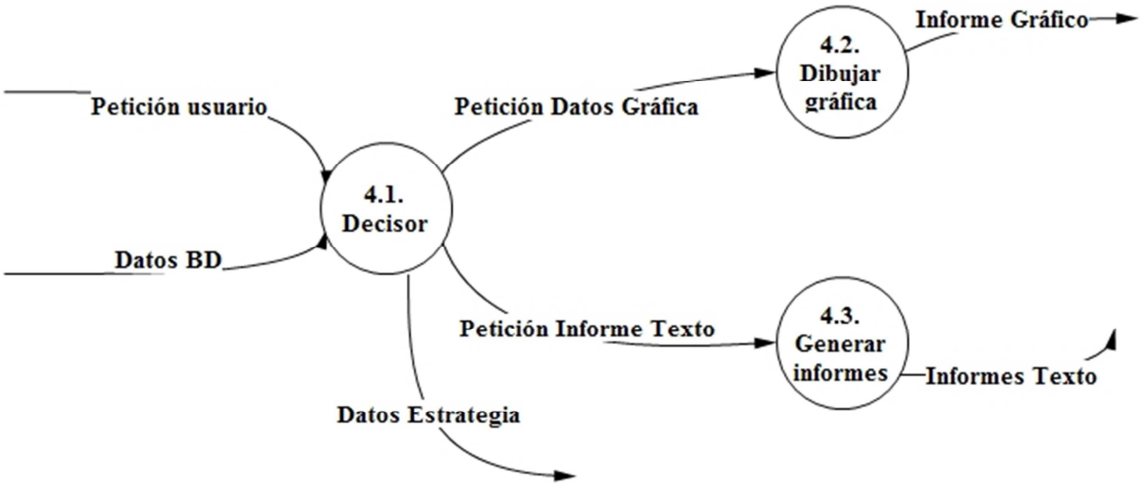

Figura 2.4: DFD de nivel 2 del proceso “Opciones usuario”

En este proceso, según la opción del usuario y recogiendo datos de la base de datos, se pueden obtener distintas salidas. Por un lado, se pueden generar datos destinados a ser ejecutado en un programa externo. Además, se puede hacer una petición para representar los datos en una gráfica o también para un informe de texto, y para cada uno de ellos se precisa un tipo de procesamiento diferente.

3. Diseño

ÍNDICE

3.1. Diseño Base de Datos
- 3.1.1. Información general
  - 3.1.1.1. SGBD
  - 3.1.1.2. SQL Server
  - 3.1.1.3. ODBC
- 3.1.2. Creación Base de Datos
  - 3.1.2.1. Modelo Entidad-Relación
  - 3.1.2.2. Modelo Relacional
- 3.1.3. Mantenimiento de la Base de Datos
  - 3.1.3.1. Microrred Maintenance Plan
  - 3.1.3.2. Differential Back Up
  - 3.1.3.3. Log Transaction Back Up

3.2. Diseño aplicación LabVIEW
- 3.2.1. Explicación general
- 3.2.2. Esquema General
- 3.2.3. Esquema General Térmico
- 3.2.4. Esquema Comunicación
- 3.2.5. Gráficas en tiempo real
- 3.2.6. Históricos
- 3.2.7. Selección de datos
- 3.2.8. FTP
- 3.2.9. Interacción con Base de Datos
- 3.2.10. Librerías creadas
  a) Global
  b) SQLToolsLLB
  c) ArrayToolsLLB
  d) DBToolsLLB
  e) FileToolsLLB
  f) ChartToolsLLB
  g) StringAndNumericToolsLLB

3.1.Diseño Base de Datos

Tras el establecimiento de todos los requisitos previos, la importancia recaía en la elección de las herramientas adecuadas que me permitieran desarrollar el proyecto de una manera adecuada y eficaz.

Es importante dentro de la microrred el correcto almacenamiento de todos los datos generados, y el número de éstos es considerablemente alto, ya que cada segundo se

generan 130 datos. Es por eso, que la cantidad y el tamaño que los datos ocupan tiene bastante relevancia.

Un diseño de la base de datos adecuado permitiría almacenar y gestionar todos los datos recogidos. Tanto el sistema gestor como la estructura de datos deberían ser potentes para ser capaces de gestionar todo esto, así como estar optimizados para tal número de transacciones.

Es por eso que la primera tarea importante dentro del diseño era la elección del gestor de la base de datos (SGBD), a partir del cual se desarrollaría todo el sistema de la base de datos.

3.1.1. Información general

3.1.1.1 SGBD

Un SGBD (Sistema Gestor de Bases de Datos) es un software que hace las veces de interfaz entre el usuario, la base de datos y las distintas aplicaciones que pueda utilizar. En este caso, resultará de intermediario entre el usuario, la base de datos a implementar y el software desarrollado en LabVIEW.

Resulta muy interesante el uso de un SGBD por varias razones:

- Independencia lógico-física: cambios en los datos no implicarán cambios en las aplicaciones.

- Rapidez: el tiempo de respuesta de un SGBD ante una consulta de complejidad media es muy reducido.

- Abstracción de los datos: los clientes usan un sistema transparente, sin importar cómo estén los datos almacenados, seguirá ofreciendo las mismas prestaciones.

3.1.1.2. SQL Server

El SGBD elegido ha sido SQL Server, concretamente la versión de 2008. A la hora de elegir el SGBD, se han tenido en cuenta diversos aspectos, y teniendo en cuenta todos ellos, SQL Server 2008 ha tenido ventaja sobre los demás. SQL Server es un software perteneciente a Microsoft que se basa en el modelo relacional. Dicho modelo es el paradigma de modelo de bases de datos más utilizado actualmente, que permite establecer relaciones entre datos almacenados por separado. Alguna de las ventajas de la utilización de SQL Server son las siguientes:

- Escalabilidad, estabilidad y seguridad: posee uno de los niveles más altos en cuanto a estas características, de modo que puede asegurar la integridad y perdurabilidad de los datos. Esto significa no solo que nuestros datos no se perderán, si no que los mismos no se corromperán, pudiendo causar un fallo grave en la base de datos.

- Entorno gráfico: El entorno gráfico del que dispone SQL Server 2008 hace muy intuitivas las tareas de manejo de las bases de datos. Es por eso, que aunque permite trabajar mediante comandos perfectamente, está adaptado con ayudas e interfaces que hacen la tarea de gestión mucho más sencilla y amigable. Es también algo importante a tener en cuenta debido a que los futuros usuarios del sistema pueden no tener conocimientos de bases de datos, y en caso de que quisieran hacer alguna consulta o modificación, con este software obtendrían mucha ayuda.

- Gratuito: gracias a MSDN y debido a que el proyecto no tenía ánimo de lucro, es posible disfrutar de la ventaja de descargar gratuitamente este software desde la página de MSDN. Esto supone una ventaja frente a otros SGBD como pueden ser Oracle, cuya licencia puede ascender a una cantidad bastante elevada.

- Información, cantidad y calidad: la primera versión de SQL Server salió al mercado en 1989. Desde entonces, se han desarrollado nuevas versiones con funcionalidades avanzadas. Durante todo este tiempo, y gracias al avance de Internet, se ha producido una cantidad inmensa de documentación sobre este software, por lo que un problema específico puede ser resuelto en un tiempo relativamente rápido gracias a toda la información existente sobre el mencionado SGBD.

- Compatibilidad con LabVIEW: Este ha sido quizás el punto clave a la hora de decidir el SGBD. En un principio, PostgreSQL o MySQL parecían una buena solución, ya que ambos son gratuitos y sencillos de utilizar. Sin embargo, a la hora de comunicarlo con LabVIEW daba muchos problemas de conexión, por lo que fue imposible seguir con esa opción. Es cierto que aunque LabVIEW está preparado para trabajar con bases de datos, no hay mucha cantidad de información al respecto. La mayoría de la información se refiere a “bases de datos” en Microsoft Access, que funcionan de un modo muy diferente a como lo puede hacer un SGBD como SQL Server, y que quedaron descartadas por su conocida ineficiencia e ineficacia. La siguiente opción fue Oracle, ya que es el SGBD sobre el que he trabajado a lo largo de la carrera universitaria. Sin embargo, también se producían los mismos errores de conexión como con los SGBD gratuitos. Así pues, al comprobar que SQL Server era capaz de comunicarse con LabVIEW correctamente, y dado también todo lo anterior, fue en efecto el SGBD elegido.

3.1.1.3. ODBC

Las siglas ODBC provienen de Open DataBase Connectivity. Es un estándar de bases de datos utilizado en entornos de Microsoft. Su objetivo es lograr una conexión transparente con la base de datos, de modo que no importe qué SGBD estemos utilizando, ya que se traducirán las consultas y ejecutarán correctamente.

Para poder conectarse a una base de datos, el SGBD debe tener un driver ODBC. En nuestro caso, SQLServer, dicho driver ya está instalado por defecto en el sistema operativo usado, Windows 7. Se puede ver un esquema del funcionamiento en la Figura 3.1.

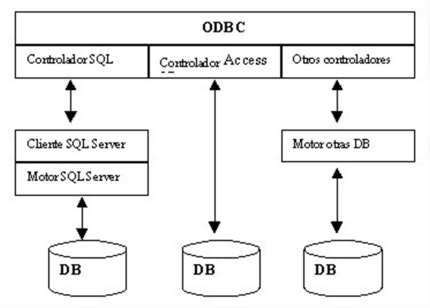

Figura 3.1: ODBC

Los orígenes de datos de ODBC son cada día menos utilizados, y se reemplazan por tecnologías como ADO.NET, ya que ODBC es tan genérica que no es capaz de obtener todo el rendimiento posible de los distintos tipos de SGBD. Sin embargo, LabVIEW funciona perfectamente con esta tecnología, ya que no está tan avanzado en cuanto a las bases de datos se refiere. Es por eso que se ha decidido usar ODBC para la conexión con la base de datos.

Para establecer la conexión entre LabVIEW y SQLServer mediante ODBC, necesitamos crear un DNS, que será el nombre con el que llamaremos a la base de datos desde la aplicación. Debido a que el sistema operativo instalado es de 64bits, no se puede crear desde el interfaz del panel de control del ordenador, ya que por defecto se crean como si de un sistema de 32 bits se tratasen. En la versión actual de Windows 7, el modo de crear el origen de datos es:

- Acceder a la ruta c:\windows\sysWOW64\odbcad32.exe
- Crear DNS de sistema
- Seleccionar SQL Server Agent 10.0
- Establecer la base de datos a conectar. En este caso, ha sido nombrada como “Microrred”, al igual que el origen de datos

De este modo, una vez dentro de la aplicación, nos bastará el literal “Microrred” para referenciar a la base de datos del mismo nombre ubicada en SQL Server.

3.1.2. Creación de la Base de Datos

La creación de la base de datos ha sido una tarea principal en la realización de este proyecto. Había que partir de la base de una base de datos bien diseñada e implementada, de modo que el resto de la aplicación pudiera ejecutarse de una manera correcta y eficiente.

La dificultar más importante en la creación de la base de datos ha sido el cambio de requisitos y la tardanza de definición de los mismos. No fue hasta dos meses después del inicio del proyecto cuando se recibió una primera versión de los datos a recoger junto con sus características. Dicha especificación ha cambiado continuamente durante la duración del proyecto, lo que ha conllevado a cambios constantes en la base de datos. No solo se han producido cambios en el número de datos, si no también en el tipo de los mismos, lo que afectaba considerablemente a las estimaciones del tamaño total de los datos. Esto ha resultado en que la versión final de los datos es considerablemente mayor y más compleja que la especificada al principio.

Sin embargo, aunque los datos hayan cambiado, no así lo ha hecho tanto la estructura de la base de datos. Se produjo algún cambio fundamental que afectó a la estructura de la misma, pero fundamentalmente se ha mantenido, lo que ha supuesto una gran ventaja a la hora de diseñarla e implementarla.

La microrred genera ininterrumpidamente 130 datos cada segundo, los cuales deben ser almacenados y tratados. Esto supone una gran cantidad de datos, y, por tanto, una gran cantidad de transacciones o inserciones en la base de datos. En sí, el número de datos a almacenar por segundo no es muy grande, pero la complejidad reside en la frecuencia con la que éstos aparecen, haciendo que en un solo día la cantidad de datos almacenados sea inmensa. Afortunadamente, el equipo proporcionado para el alojamiento de la base de datos no tiene grandes restricciones respecto al tamaño, ya que dispone de un disco duro de 1TB, con posibilidad de ampliación.

Salvado el problema del espacio, a la hora de diseñar la base de datos, habría que ser cuidadoso para que las inserciones fuesen lo más rápidas y eficientes posibles. Se estableció como requisito la no pérdida de los datos, por lo tanto, habrá que diseñar un mecanismo que interactúe con la base de datos al mismo tiempo que se generan éstos, de modo que no se pierda ninguno de ellos.

Además, se requiere que una parte de la aplicación sea capaz de rescatar esos datos, por lo que, aunque mucho menos frecuentes que las inserciones, las consultas también estarán presentes y deberán ser tenidas en cuenta, ya que devolverán grandes cantidades de datos, y todos ellos deben ser manejados mediante una consulta y gestión adecuados.

Los datos son enviados en un array de dobles. Las siguientes tablas (Figura 3.2 y Figura 3.3) muestran la disposición de esos datos. Por cada uno de los dispositivos, se muestra el módulo a través del cual está conectado, el código de la estación, el número de

variables, el nombre de las mismas, el rango del dato, la unidad de medida, los decimales, y, por último, la posición del array en la que se encuentra.

Los rangos se han establecido por tipo de dato o por rango de valores. Hay que tener en cuenta que es un campo informativo, ya que, en el array, todos los datos serán de tipo doble. Cabe mencionar que FRE+CAL+ACS no corresponde a ningún dispositivo físico, si no que se trata de datos simulados.

Microrred	Modulo	Estacion	Nº Variables	Variables	Rango	Unidad	Decimales	Posicion
-	-	-	1	Fecha del dato	-	-	-	0
RED	CVMk2	9600 (1)	10	Bit error	entero		-	2
				V1	0-500	V	2	3
				V2	0-500	V	2	4
				V3	0-500	V	2	5
				I1	0-150	A	2	6
				I2	0-150	A	2	7
				I3	0-150	A	2	8
				Pt	0-50	kW	3	9
				Qt	0-50	Var	0	10
				freq	0-60	Hz	2	11
Grupo diesel	NRG96	9601 (12)	9	Bit error	entero	-	-	13
				V1	0-500	V	1	14
				V2	0-500	V	1	15
				V3	0-500	V	1	16
				I1	0-150	A	1	17
				I2	0-150	A	1	18
				I3	0-150	A	1	19
				Pt	0-50	kW	3	20
				freq	0-60	Hz	1	21
Cargas no criticas	NRG96	9602 (22)	10	Bit error	entero		-	23
				V1	0-500	V	1	24
				V2	0-500	V	1	25
				V3	0-500	V	1	26
				I1	0-150	A	1	27
				I2	0-150	A	1	28
				I3	0-150	A	1	29
				Pt	0-50	kW	3	30
				Qt	0-50	kVar	3	31
				freq	0-60	Hz	1	32
Baterias	DH96	9603 (33)	4	Bit error	entero	-	-	34
				V	0-500	V	1	35
				I	0-150	A	1	36
				Pt	0-10	kW	3	37
Ultracondensadores	DH96	9604 (38)	4	Bit error	entero	-	-	39
				V	0-500	V	1	40
				I	0-150	A	1	41
				Pt	0-10	kW	3	42
Hidrogeno	DH96	9605 (43)	4	Bit error	entero	-	-	44
				V	0-500	V	1	45
				I	0-150	A	1	46
				Pt	0-10	kW	3	47
Fotovoltaica	DH96	9606 (48)	4	Bit error	entero	-	-	49
				V	0-500	V	1	50
				I	0-150	A	1	51
				Pt	0-10	kW	3	52
Hibrido cargador	Ingeteam	9800 (53)	30	Bit error	entero		-	54
				Vbat	0-500	V	0	55
				Ibat	0-150	V	1	56
				SOC	0-100	V	0	57
				VPV	0-500	A	0	58
				IPV	0-150	A	0	59

Figura 3.2: Tabla de variables de los módulos (1)

Microrred	Modulo	Estacion	Nº Variables	Variables	Rango	Unidad	Decimales	Posicion
Hibrido inversor	Ingeteam		15	V1	0-500	V	0	60
				V2	0-500	V	0	61
				V3	0-500	V	0	62
				Pt1	0-150	A	1	63
				Pt2	0-150	A	1	64
				Pt3	0-150	A	1	65
				Diesel V1	0-500	V	0	66
				Diesel V2	0-500	V	0	67
				Diesel V3	0-500	V	0	68
				Diesel Hz1	0-60	Hz	0	69
				Diesel Hz2	0-60	Hz	0	70
				Diesel Hz3	0-60	Hz	0	71
Hibrido Eolica	Ingeteam		7	Pt1	0-50	kW	3	72
				Pt2	0-60	Hz	0	73
				Pt3	0-60	Hz	0	74
				Hz1	0-60	Hz	0	75
				Hz2	0-130	ºC	0	76
				Hz3	0-1000	V	0	77
Cargas criticas	PR300	9700 (78)	10	Bit error	entero		-	78
				V1	0-500	V	2	79
				V2	0-500	V	2	80
				V3	0-500	V	2	81
				I1	0-150	A	2	82
				I2	0-150	A	2	83
				I3	0-150	A	2	84
				Qt	0-50	kVar	3	85
				freq	0-60	Hz	2	86
Salida Inversor	PR300	9701 (89)	10	Bit error	entero		-	90
				V1	0-500	V	2	91
				V2	0-500	V	2	92
				V3	0-500	V	2	93
				I1	0-150	A	2	94
				I2	0-150	A	2	95
				I3	0-150	A	2	96
				Pt	0-50	kW	3	97
				Qt	0-50	kVar	3	98
				freq	0-60	Hz	2	99
Aerogenerador	PR300	9702 (100)	10	Bit error	entero		-	101
				V1	0-80	V	2	102
				V2	0-80	V	2	103
				V3	0-80	V	2	104
				I1	0-150	A	2	105
				I2	0-150	A	2	106
				I3	0-150	A	2	107
				Pt	0-10	kW	3	108
				Qt	0-10	kVar	3	109
				freq	0-100	Hz	2	110
Meteorologicas	PXI-6238	9900 (111)	9	Tª exterior	_-20:80	ºC	2	112
				Celula FV	0-2500	W/m2	1	113
				Tª Celula	_-20:80	ºC	2	114
				Anemometro celula	_0:200	km/h	1	115
				Anemometro 1	_0:200	km/h	1	116
				Anemometro 2	_0:200	km/h	1	117
				Tª Sotano	_-20-80	ºC	2	118
				Tª Baterias	_-20-80	ºC	2	119
				Rele 1	bool	-	-	120
				Rele 2	bool	-	-	121
				Rele 3	bool	-	-	122
				Rele 4	bool	-	-	123
FRE+CAL+ACS				Pt Caldera	real	kW		128
				Pt CO	real	kW		129
				Tª Caldera	real	ºC		130

Figura 3.3: Tabla de variables de los módulos (2)

3.1.2.1.Modelo Entidad – Relación

El Modelo Entidad-Relación es un modelo utilizado para representar gráficamente la estructura lógica de una base de datos. Sus elementos principales son las entidades y las relaciones entre entidades.

Una entidad es un objeto del que se desea almacenar información. Tendrá una serie de campos o atributos con los datos oportunos. Una relación indica una unión lógica entre distintas entidades.

En el diseño de esta base de datos, en concreto del Modelo Entidad/Relación, dado que se consideró que cumplía con unas características especiales, se pidió opinión a la profesora de la UPNA Victoria Mohedano, Dra en Matemáticas y experta en bases de datos. De esta manera el diseño resultó del modo especificado a continuación.

En nuestro caso, queda claro que las entidades son los distintos dispositivos de los que obtendremos los datos. Así pues, en primera instancia, podríamos establecer como entidades las siguientes:

-Aerogenerador

-Baterías

-Cargas Críticas

-Cargas no críticas

-Fotovoltaica

-FRE+CAL+ACS

-Grupo Diesel

-Híbrido Cargador

-Híbrido Eólica

-Híbrido Inversor

-Hidrógeno

-Meteorológicas

-RED

-Salida Inversor

-Ultracondensadores

Sin embargo, el fundamento de una base de datos es el almacenamiento de datos únicos y distinguibles, relacionados entre sí. Pero los datos de los distintos módulos no son únicos, ya que, siendo tal la cantidad de datos generados, es previsible que, por ejemplo, en el dispositivo de Baterías, se repita en algún momento la tupla V-I-Pt. Por otro lado, tampoco se encuentran fácilmente relaciones entre los datos, ya que los dispositivos actúan por separado y son independientes unos de otros, es decir, los datos que ofrece un dispositivo no cambian en función de los que ofrezca otro.

Pero, sin unicidad de datos ni relación entre ellos, es imposible establecer una base de datos relacional. Así pues, hay que "forzar" a los datos a que se relacionen. Las razones para "forzar" esta relación son la facilidad futura de ampliación de la base de datos, ya que, según la estructura a establecer, añadir otro dispositivo será trivial. Además, interesa establecer relaciones en caso de que en algún momento haya dispositivos relacionados, que dependan uno de otro. La solución propuesta es la siguiente:

Si bien es cierto que la tupla V-I-Pt se puede repetir, cambiará el momento en el que suceda. Es decir, puede que unos valores x-y-z se repitan exactamente al cabo de unas horas, o minutos, pero habrá cambiado el momento en el que se produzcan. Esa es la base para la unicidad de los datos. Cada dispositivo tendrá, pues, unos campos o atributos adicionales a los expuestos en las tablas anteriores. Se tratará de la fecha y la hora en la que se produzcan dichos datos. Esa información la podemos recoger de la primera posición del array que se recibe.

Una vez añadidos esos campos, es más fácil ver la relación entre las distintas tablas. Crearemos una nueva tabla llamada Momento que almacenará todas las fechas y horas en las que se registren datos. Cada uno de los dispositivos se relacionará con este Momento, ya que cada uno de los datos se producirá en una fecha y hora conocidas.

Se puede decir entonces, que los distintos dispositivos se relacionarán con un Momento dado. Así pues, se define que todos los dispositivos son entidades débiles de Momento. Una entidad débil se define como una entidad cuyos datos no son únicos sin participar en la relación.

Resumiendo: Momento tendrá los datos de fecha y hora, y los demás dispositivos se relacionarán con Momento para agregar a sus datos la fecha y la hora, de modo que se conviertan en datos únicos.

Así pues, el esquema entidad-relación (Figura 3.4) es el siguiente:

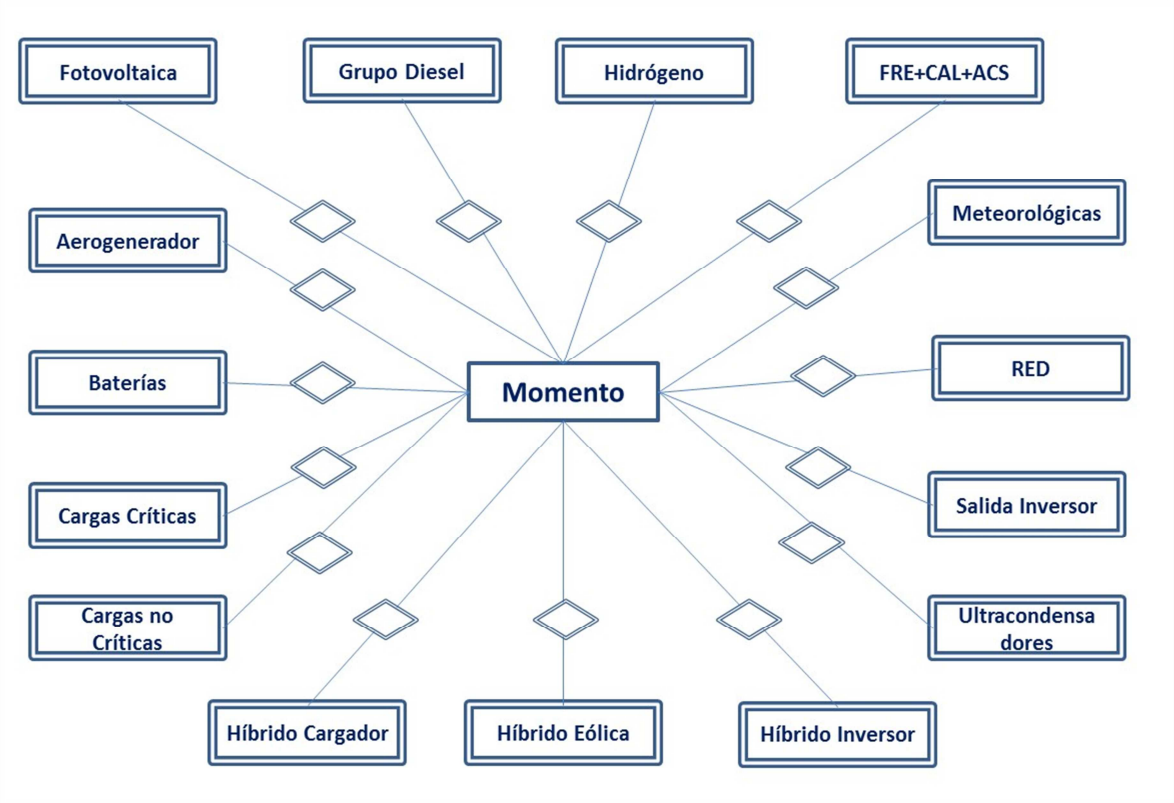

Figura 3.4: Esquema entidad-relación

Las entidades débiles son representadas como doble rectángulo, y las relaciones débiles (en las que participen las entidades débiles) como un doble rombo.

3.1.2.2.Modelo Relacional

El Modelo Relacional propuesto por Codd considera la base de datos como si de un conjunto de relaciones se tratase. Cada una de estas relaciones se puede considerar como una tabla compuesta por una serie de registros (filas) y campos (columnas).

De este modo, convertiremos el modelo Entidad-Relación desarrollado anteriormente en el modelo relacional, lo cual nos permitirá ver el esquema de la base de datos, de forma que pueda ser implementado en SQL Server.

Cada una de las entidades será una relación, y los campos corresponderán a los distintos tipos de datos que almacenaremos de cada relación.

Para cada una de las relaciones se especificará sus atributos, el tiempo de dato de los mismos, su clave primaria, su clave ajena en caso de que exista y la secuencia de creación de la relación.

Cabe especificar que ninguno de los valores en la base de datos puede ser nulo, ya que en todo momento llegarán todos los datos con sus valores correspondientes. Además, la base de datos a crear tendrá como nombre “Microrred”.

Se puede ver el esquema del Modelo Relacional en la Figura 3.5.

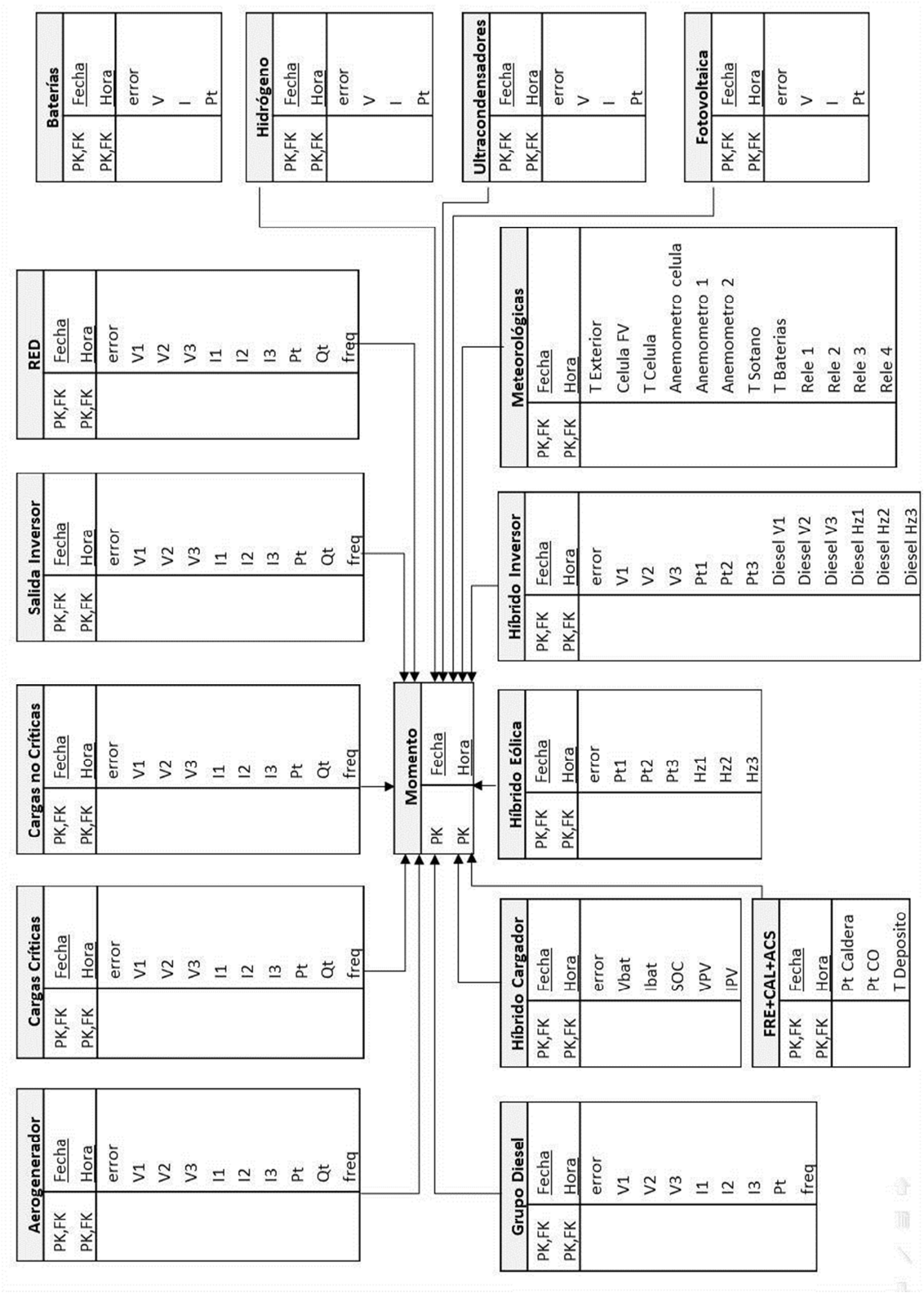

Figura 3.5: Esquema del modelo relacional

a) Momento

Descripción: Esta relación almacena la información de los tiempos en los que se generan los datos.

Atributo	Tipo de dato	Clave
Fecha	Date	Primaria
Hora	Time(0)	Primaria

Código de generación:

```
CREATE TABLE Momento (
  Fecha        date          not null,
  Hora         time(0)       not null,
PRIMARY KEY (Fecha, Hora)
)
```

b) Aerogenerador

Descripción: Esta relación almacena la información del dispositivo “Aerogenerador”. La variable de error valdrá 0 si no hay error.

Atributo	Tipo de dato	Clave
Fecha	date	Primaria, Foránea
Time	Time(0)	Primaria, Foránea
error	real	
V1	real	
V2	real	
V3	real	
I1	real	
I2	real	
I3	real	
Pt	real	
Qt	real	
freq	real	

Código de generación:

```
CREATE TABLE Aerogenerador(
  Fecha        date not null,
  Hora         time(0) not null,
  error        real not null,
  V1           real not null,
  V2           real not null,
```

```
  V3            real not null,
  I1            real not null,
  I2            real not null,
  I3            real not null,
  Pt            real not null,
  Qt            real not null,
  freq          real not null,
FOREIGN KEY (Fecha, Hora) REFERENCES Momento (Fecha,Hora),
PRIMARY KEY (Fecha,Hora)
)
```

c) Baterías

Descripción: Esta relación almacena la información del dispositivo “Baterías”. La variable de error valdrá 0 si no hay error.

Atributo	**Tipo de dato**	**Clave**
Fecha	date	Primaria, Foránea
Time	Time(0)	Primaria, Foránea
error	real	
V	real	
I	real	
Pt	real	

Código de generación:

```
CREATE TABLE Baterias(
  Fecha date not null,
  Hora   time(0) not null,
  error   real not null,
  V       real not null,
  I        real not null,
  Pt       real not null,
FOREIGN KEY (Fecha, Hora) REFERENCES Momento (Fecha,Hora),
PRIMARY KEY(Fecha,Hora)
)
```

d) Cargas críticas

Descripción: Esta relación almacena la información del dispositivo “Cargas Críticas”. La variable de error valdrá 0 si no hay error.

Atributo	Tipo de dato	Clave
Fecha	date	Primaria, Foránea
Time	Time(0)	Primaria, Foránea
error	real	
V1	real	
V2	real	
V3	real	
I1	real	
I2	real	
I3	real	
Pt	real	
Qt	real	
freq	real	

Código de generación:

```
CREATE TABLE Cargas_Criticas(
  Fecha        date not null,
  Hora         time(0) not null,
  error        real not null,
  V1           real not null,
  V2           real not null,
  V3           real not null,
  I1           real not null,
  I2           real not null,
  I3           real not null,
  Pt           real not null,
  Qt           real not null,
  freq         real not null,
FOREIGN KEY (Fecha, Hora) REFERENCES Momento (Fecha,Hora),
PRIMARY KEY (Fecha,Hora)
)
```

e) Cargas no críticas

Descripción: Esta relación almacena la información del dispositivo “Cargas No Críticas”. La variable de error valdrá 0 si no hay error.

Atributo	Tipo de dato	Clave
Fecha	date	Primaria, Foránea
Time	Time(0)	Primaria, Foránea
error	real	
V1	real	
V2	real	
V3	real	
I1	real	
I2	real	
I3	real	
Pt	real	
Qt	real	
freq	real	

Código de generación:

```
CREATE TABLE Cargas_No_Criticas(
  Fecha        date not null,
  Hora         time(0) not null,
  error        real not null,
  V1           real not null,
  V2           real not null,
  V3           real not null,
  I1           real not null,
  I2           real not null,
  I3           real not null,
  Pt           real not null,
  Qt           real not null,
  freq         real not null,
FOREIGN KEY (Fecha, Hora) REFERENCES Momento (Fecha,Hora),
PRIMARY KEY (Fecha,Hora)
)
```

f) Fotovoltaica

Descripción: Esta relación almacena la información del dispositivo “Fotovoltaica”. La variable de error valdrá 0 si no hay error.

Atributo	Tipo de dato	Clave
Fecha	date	Primaria, Foránea
Time	Time(0)	Primaria, Foránea
error	real	
V	real	
I	real	
Pt	real	

Código de generación:

```
CREATE TABLE Fotovoltaica(
  Fecha date not null,
  Hora   time(0) not null,
  error   real not null,
  V       real not null,
  I        real not null,
  Pt       real not null,
FOREIGN KEY (Fecha, Hora) REFERENCES Momento (Fecha,Hora),
PRIMARY KEY(Fecha,Hora)
)
```

g) FRE+CAL+ACS

Descripción: Esta relación almacena la información de los datos generados correspondientes a FRE+CAL+ACS. Dado que son datos simulados, no contienen variable de error y se considerarán siempre válidos.

Atributo	Tipo de dato	Clave
Fecha	date	Primaria, Foránea
Time	Time(0)	Primaria, Foránea
Pt_Caldera	real	
Pt_CO	real	
T_Deposito	real	

Código de generación:

```
CREATE TABLE FRE_CAL_ACS(
  Fecha                 date not null,
  Hora                  time(0) not null,
```

```
  Pt_Caldera        real not null,
  Pt_CO             real not null,
  T_Deposito        real not null,
FOREIGN KEY (Fecha, Hora) REFERENCES Momento (Fecha,Hora),
PRIMARY KEY (Fecha,Hora)
)
```

h) Grupo Diesel

Descripción: Esta relación almacena la información del dispositivo “Grupo Diesel”. La variable de error valdrá 0 si no hay error.

Atributo	**Tipo de dato**	**Clave**
Fecha	date	Primaria, Foránea
Time	Time(0)	Primaria, Foránea
error	real	
V1	real	
V2	real	
V3	real	
I1	real	
I2	real	
I3	real	
Pt	real	
freq	real	

Código de generación:

```
CREATE TABLE Grupo_Diesel(
  Fecha         date not null,
  Hora          time(0) not null,
  error         real not null,
  V1            real not null,
  V2            real not null,
  V3            real not null,
  I1            real not null,
  I2            real not null,
  I3            real not null,
  Pt            real not null,
  freq          real not null,
FOREIGN KEY (Fecha, Hora) REFERENCES Momento (Fecha,Hora),
PRIMARY KEY (Fecha,Hora)
)
```

i) Híbrido Cargador

Descripción: Esta relación almacena la información del dispositivo “Híbrido Cargador”. La variable de error valdrá 0 si no hay error.

Atributo	**Tipo de dato**	**Clave**
Fecha	date	Primaria, Foránea
Time	Time(0)	Primaria, Foránea
error	real	
Vbat	real	
Ibat	real	
SOC	real	
VPV	real	
IPV	real	

Código de generación:

```
CREATE TABLE Hibrido_Cargador(
  Fecha        date not null,
  Hora         time(0) not null,
  error        real not null,
  Vbat         real not null,
  Ibat         real not null,
  SOC          real not null,
  VPV          real not null,
  IPV          real not null,
FOREIGN KEY (Fecha, Hora) REFERENCES Momento (Fecha,Hora),
PRIMARY KEY (Fecha,Hora)
)
```

j) Híbrido Eólica

Descripción: Esta relación almacena la información del dispositivo “Híbrido Eólica”. La variable de error valdrá 0 si no hay error.

Atributo	**Tipo de dato**	**Clave**
Fecha	date	Primaria, Foránea
Time	Time(0)	Primaria, Foránea
error	real	
Pt1	real	
Pt2	real	
Pt3	real	
Hz1	real	
Hz2	real	
Hz3	real	

Código de generación:

```
CREATE TABLE Hibrido_Eolica(
  Fecha        date not null,
  Hora         time(0) not null,
  error        real not null,
  Pt1          real not null,
  Pt2          real not null,
  Pt3          real not null,
  Hz1          real not null,
  Hz2          real not null,
  Hz3          real not null,
FOREIGN KEY (Fecha, Hora) REFERENCES Momento (Fecha,Hora),
PRIMARY KEY (Fecha,Hora)
)
```

k) Híbrido Inversor

Descripción: Esta relación almacena la información del dispositivo “Híbrido Inversor”. La variable de error valdrá 0 si no hay error.

Atributo	Tipo de dato	Clave
Fecha	date	Primaria, Foránea
Time	Time(0)	Primaria, Foránea
error	real	
V1	real	
V2	real	
V3	real	
Pt1	real	
Pt2	real	
Pt3	real	
Diesel V1	real	
Diesel V2	real	
Diesel V3	real	
Diesel Hz1	real	
Diesel Hz2	real	
Diesel Hz3	real	

Código de generación:

```
CREATE TABLE Hibrido_Inversor(
  Fecha            date not null,
  Hora             time(0) not null,
  error            real not null,
  V1               real not null,
  V2               real not null,
  V3               real not null,
  Pt1              real not null,
  Pt2              real not null,
  Pt3              real not null,
  Diesel_V1        real not null,
  Diesel_V2        real not null,
  Diesel_V3        real not null,
  Diesel_Hz1   real not null,
  Diesel_Hz2   real not null,
  Diesel_Hz3   real not null,
FOREIGN KEY (Fecha, Hora) REFERENCES Momento (Fecha,Hora),
PRIMARY KEY (Fecha,Hora)
)
```

l) Hidrógeno

Descripción: Esta relación almacena la información del dispositivo “Hidrógeno”. La variable de error valdrá 0 si no hay error.

Atributo	Tipo de dato	Clave
Fecha	date	Primaria, Foránea
Time	Time(0)	Primaria, Foránea
error	real	
V	real	
I	real	
Pt	real	

Código de generación:

```
CREATE TABLE Hidrógeno(
  Fecha date not null,
  Hora   time(0) not null,
  error   real not null,
  V       real not null,
  I        real not null,
  Pt       real not null,
FOREIGN KEY (Fecha, Hora) REFERENCES Momento (Fecha,Hora),
PRIMARY KEY(Fecha,Hora)
)
```

m) Meteorológicas

Descripción: Esta relación almacena la información a los dispositivos Meteorológicos. No hay variable de error general, si no que cada una de las variables representará estado de error en el caso en el que valga -100.

Atributo	**Tipo de dato**	**Clave**
Fecha	date	Primaria, Foránea
Time	Time(0)	Primaria, Foránea
T_Exterior	real	
Celula_FV	real	
T_Celula	real	
Anemometro_Celula	real	
Anemometro_1	real	
Anemometro_2	real	
T_Sotano	real	
T_Baterias	real	
Rele_1	real	
Rele_2	real	
Rele_3	real	
Rele_4	real	

Código de generación:

```
CREATE TABLE Meteorologicas(
  Fecha date not null,
  Hora   time(0) not null,
  T_Exterior              real not null,
  Celula_FV                    real not null,
  T_Celula                     real not null,
  Anemometro_Celula       real not null,
  Anemometro_1            real not null,
  Anemometro_2            real not null,
  T_Sotano                     real not null,
  T_Baterias                  real not null,
  Rele_1                       real not null,
  Rele_2                       real not null,
  Rele_3                       real not null,
  Rele_4                       real not null,
FOREIGN KEY (Fecha, Hora) REFERENCES Momento (Fecha,Hora),
PRIMARY KEY(Fecha,Hora)
)
```

n) RED

Descripción: Esta relación almacena la información del dispositivo “RED”. La variable de error valdrá 0 si no hay error.

Atributo	Tipo de dato	Clave
Fecha	date	Primaria, Foránea
Time	Time(0)	Primaria, Foránea
error	real	
V1	real	
V2	real	
V3	real	
I1	real	
I2	real	
I3	real	
Pt	real	
Qt	real	
freq	real	

Código de generación:

```
CREATE TABLE RED(
  Fecha        date not null,
  Hora         time(0) not null,
  error        real not null,
  V1           real not null,
  V2           real not null,
  V3           real not null,
  I1           real not null,
  I2           real not null,
  I3           real not null,
  Pt           real not null,
  Qt           real not null,
  freq         real not null,
FOREIGN KEY (Fecha, Hora) REFERENCES Momento (Fecha,Hora),
PRIMARY KEY (Fecha,Hora)
)
```

o) Salida_Inversor

Descripción: Esta relación almacena la información del dispositivo “Salida Inversor”. La variable de error valdrá 0 si no hay error.

Atributo	Tipo de dato	Clave
Fecha	date	Primaria, Foránea
Time	Time(0)	Primaria, Foránea
error	real	
V1	real	
V2	real	
V3	real	
I1	real	
I2	real	
I3	real	
Pt	real	
Qt	real	
freq	real	

Código de generación:

```
CREATE TABLE Salida_Inversor(
  Fecha        date not null,
  Hora         time(0) not null,
  error        real not null,
  V1           real not null,
  V2           real not null,
  V3           real not null,
  I1           real not null,
  I2           real not null,
  I3           real not null,
  Pt           real not null,
  Qt           real not null,
  freq         real not null,
FOREIGN KEY (Fecha, Hora) REFERENCES Momento (Fecha,Hora),
PRIMARY KEY (Fecha,Hora)
)
```

p) Ultracondensadores

Descripción: Esta relación almacena la información del dispositivo “Ultracondensadores”. La variable de error valdrá 0 si no hay error.

Atributo	Tipo de dato	Clave
Fecha	date	Primaria, Foránea
Time	Time(0)	Primaria, Foránea
error	real	
V	real	
I	real	
Pt	real	

Código de generación:

```
CREATE TABLE Ultracondensadores(
  Fecha date not null,
  Hora   time(0) not null,
  error   real not null,
  V       real not null,
  I        real not null,
  Pt       real not null,
FOREIGN KEY (Fecha, Hora) REFERENCES Momento (Fecha,Hora),
PRIMARY KEY(Fecha,Hora)
)
```

3.1.3. Mantenimiento de la Base de Datos

Suponiendo implementada la base de datos que se ha diseñado en los apartados anteriores, resulta de importancia diseñar un plan de mantenimiento de la base de datos. La elaboración de un plan de mantenimiento nos asegurará de que la base de datos está optimizada y no contiene incoherencias, así como de que se realizarán copias de seguridad periódicas de modo que se puedan recuperar los datos en caso de pérdida.

Aunque las tareas de optimización y comprobación de coherencia de datos son importantes, no cabe duda que el resguardo de los datos en copias de seguridad es esencial. De este modo, ante cualquier fallo del software, o si deseamos migrar la base de datos a otro servidor, será posible recuperar toda la estructura de la base de datos, junto con todos los datos almacenados.

Para la creación de los planes de mantenimiento se ha usado el asistente de creación de planes de mantenimiento proporcionado por SQL Server, al que se puede acceder desde la interfaz del programa, siguiendo estos pasos:

-Expandir el árbol

-Management

-Maintenance Plans (click derecho)

-Maintenance Plan Wizard

Para acceder a los planes de mantenimiento que se van a describir, se puede hacer desde la interfaz gráfica de SQL Server, siguiendo estos pasos:

-Expandir el árbol

-Management

-Maintenance Plan

Dado el tamaño de la base de datos actual, y la importancia y cantidad de sus datos, se han establecido tres planes de mantenimiento de la base de datos, que trabajan complementándose, de modo que se consiga la máxima optimización de la misma. Para cada plan, se establece una breve descripción, su calendario de ejecución, un esquema de ejecución, las tareas que lo comprenden y el código en Transact-SQL(T-SQL) correspondiente en las tareas en las que no se haya generado automáticamente.

Los nombres de las tareas están escritos en inglés, ya que la versión de SQL Server 2008 está en ese mismo idioma, por lo que es así como lo encontraremos si accedemos desde la interfaz gráfica.

Antes de detallar los planes de mantenimiento, cabe establecer la diferencia entre los distintos tipos de copias de seguridad, con el fin de entender el porqué de la elección escogida.

Hay tres tipos básicos de copias de seguridad de datos en una base de datos: completa (FULL), incremental (differential) y del log de transacciones (log transaction).

Siempre tiene que existir una copia completa de la base de datos. Este tipo de copia almacena toda la información de la base de datos, y es por eso que es el tipo de copia con más información, pero también la que más tarda en realizarse.

La copia incremental no guarda todos los datos de la base de datos, solo aquellos que han sido modificados desde la última copia de seguridad (ya sea completa o incremental). Si nunca antes se ha ejecutado una copia de seguridad, la primera copia incremental actuaría como una copia completa.

La copia del log de transacciones realiza una copia del registro de transacciones, que almacena todas las transacciones y modificaciones que se han realizado en la base de datos. Es decir, aunque no tenga datos, tendrá las sentencias necesarias, para que, en caso de ejecutarse, se almacenen esos datos en la base de datos.

Así pues, las copias de seguridad incrementales complementan a las copias de seguridad completas. La copia del log de transacciones complementa a la copia de seguridad incremental, ya que contendrá todas las sentencias necesarias para generar los datos que se hayan almacenado desde la última copia incremental (o completa).

Así pues, la estrategia es la siguiente:

-Se realizan copias de seguridad completas una vez por semana.

-Se realizan copias de seguridad incrementales una vez por día.

-Se realizan copias de seguridad del fichero de transacciones una vez por cada 8 horas.

La especificación de los planes es la siguiente:

3.1.3.1.Microrred Maintenance Plan

Este plan es el plan general de mantenimiento de la base de datos. Comprende una serie de tareas relacionadas entre sí y dispuestas secuencialmente, de modo que se consiga el máximo rendimiento. El esquema de funcionamiento se puede ver en la Figura 3.6.

Calendario de ejecución: Todos los domingos a las 00:30:00h

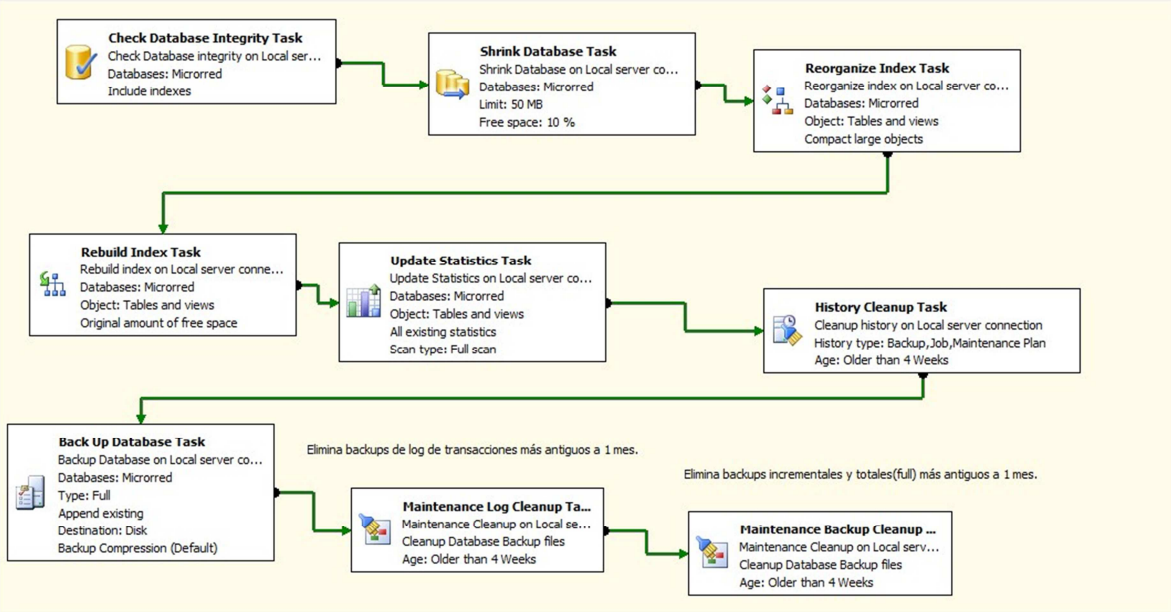

Figura 3.6: Esquema del plan de mantenimiento "Microrred Maintenance Plan"

i. Check Database Integrity Task

La tarea "Chequeo de la integridad de la base de datos" realiza comprobaciones de coherencia interna de las páginas de datos y el índice dentro de la base de datos.

ii. Shrink Database Task

La tarea "Reducir base de datos" reduce el espacio de disco consumido por los archivos de base de datos y log (registro) mediante la eliminación de datos vacíos y las páginas de log.

En este caso, se ha establecido que esta tarea se ejecuta siempre que el tamaño de la base de datos sea mayor a 50MB. La cantidad de espacio libre que permanecerá después de la reducción será un 10%, y el espacio que liberemos será devuelto al sistema operativo.

iii. Reorganize Index Task

La tarea "Reorganizar índices" desfragmenta y compacta índices agrupados (clustered) y no agrupados en tablas y vistas. Esto mejorará el índice de rendimiento de recorrido.

iv. Rebuild Index Task

La tarea "Reconstrucción de índices" reorganiza los datos en las páginas de datos e índices reconstruyendo los índices. Esto mejora el rendimiento en escaneo de índices y las búsquedas. Esta tarea también optimiza la distribución de datos y espacio libre en las páginas de índices, permitiendo un crecimiento más rápido en el futuro.

v. Update Statistics

La tarea "Actualizar estadísticas" asegura que el optimizador de consultas tiene información actualizada sobre la distribución de los valores de los datos en las tablas. Esto permite al optimizador tomar mejores decisiones acerca de las estrategias de acceso a datos.

vi. History Cleanup Task

La tarea "Limpieza del historial" elimina datos históricos acerca de copias de seguridad y restauración, del Agente SQL Server y las operaciones del plan de mantenimiento.

Se eliminarán los datos del historial que sean más antiguos que 1 mes.

vii. Back Up Database Task

Esta tarea realiza copias de seguridad totales (full) de la base de datos Microrred en el destino C:\Program Files\Microsoft SQL Server\ MSSQL10.MSSQLSERVER \MSSQL\ Backup.

viii. Maintenance Log Cleanup Task

Esta tarea de "Limpieza de mantenimiento" elimina backups del log de transacciones más antiguos a un mes.

ix. Maintenance Backup Cleanup Task

Esta tarea de "Limpieza de mantenimiento" elimina las copias de seguridad incrementales y totales con más antiguos a un mes.

3.1.3.2.Differential Back Up

Este segundo plan de mantenimiento contiene solo una tarea, y se encargará de realizar las copias de seguridad incrementales de la base de datos. El esquema de funcionamiento se puede ver en la Figura 3.7.

Calendario de ejecución: Todos los días a las 00:15:00h

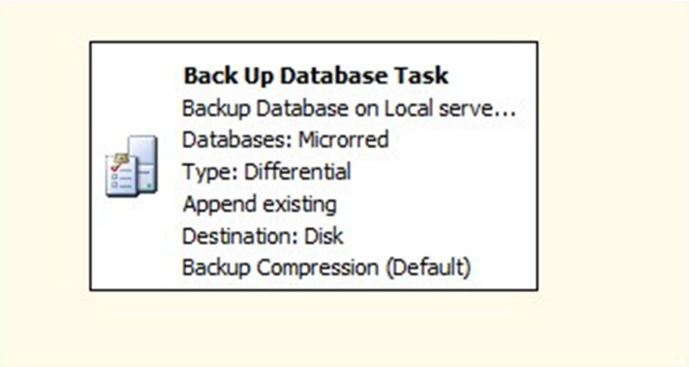

Figura 3.7: Esquema del plan de mantenimiento "Differential Back Up"

i. Back Up Database Task

Esta tarea realiza copias de seguridad incrementales (differential) de la base de datos Microrred en el destino C:\Program Files\Microsoft SQL Server\ MSSQL10.MSSQLSERVER \MSSQL\ Backup.

3.1.3.3.Log Transaction Back Up

Este tercer plan de mantenimiento contiene tres tareas, y se encargará de realizar las copias de seguridad del fichero de transacciones de la base de datos. El esquema de funcionamiento se puede ver en la Figura 3.8.

Calendario de ejecución: Todos los días cada 8h.

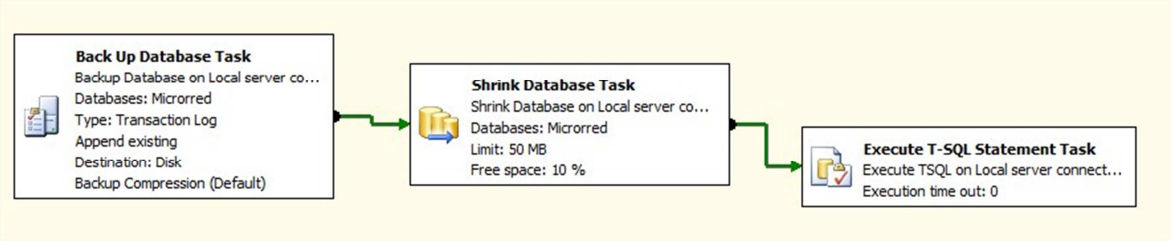

Figura 3.8: Esquema del plan de mantenimiento "Log Transaction Back Up"

i. Back Up Database Task

Esta tarea realiza copias de seguridad del fichero de transacciones (transaction log) de la base de datos Microrred en el destino C:\Program Files\Microsoft SQL Server\ MSSQL10.MSSQLSERVER \MSSQL\ Backup.

ii. Shrink Database Task

La tarea "Reducir base de datos" reduce el espacio de disco consumido por los archivos de base de datos y log (registro) mediante la eliminación de datos vacíos y las páginas de log.

En este caso, se ha establecido que esta tarea se ejecuta siempre que el tamaño de la base de datos sea mayor a 50MB. La cantidad de espacio libre que permanecerá después de la reducción será un 10%, y el espacio que liberemos será devuelto al sistema operativo.

iii. Execute T-SQL Statement Task

En esta tarea se ha definido un código propio de T-SQL, cuya función es reducir al máximo posible el fichero de log, ya que la tarea anterior puede no hacerlo.
T-SQL:
USE Microrred

GO

CHECKPOINT

DBCC SHRINKFILE (Microrred_log, 200)

DBCC SHRINKFILE (Microrred_log, 200)

3.2.Diseño LabVIEW

3.2.1. Explicación general

Consideraciones iniciales: Las aplicaciones diseñadas en LabVIEW son generadas con una extensión ".vi", de modo que será común a lo largo de este documento referirse indistintamente a programa o "vi", ya que, en esencia, son lo mismo. Los subprogramas creados en LabVIEW son llamados "subVIs". En LabVIEW, el interfaz con el usuario es llamado "Panel Frontal" y el interfaz con el programador, "diagrama de bloques".

La aplicación se ejecutará en un ordenador de propósito general con una triple pantalla, con unas medidas de 1928x1080px. Por tanto, debe estar optimizada de modo que aproveche lo más posible esta característica. Así pues, debe ser amigable con el usuario, y con un diseño atractivo y visual en el que se pueden consultar cómodamente todos los datos deseados.

En todo momento se ha tratado de realizar aplicaciones independientes y modularizadas, de modo que sea sencillo modificarlas, además de que puedan funcionar independientes unas de otras. Sin embargo, por cuestiones de eficiencia del rendimiento, alguna de las aplicaciones depende una de otra, ya que se ha considerado que siempre se ejecutarán simultáneamente. Un ejemplo es el esquema general y las gráficas en tiempo real. Sin embargo, los programas con funcionalidades muy diferenciadas, como puede ser el recolector de ficheros FTP, funcionan totalmente independientes de los demás.

Se parte de la base de que llega un array con los datos deseados, y serán estos datos los que se usen para rellenar todas las pantallas, y ofrecer toda la información disponible al usuario. Se presupone que los datos enviados son los correctos, ya que no entra dentro del ámbito de este proyecto la conexión del PXi con los distintos dispositivos y el ordenador. Sin embargo, si existe un valor que indicará si se ha perdido conexión entre el PXi y el PC, para tomar las medidas oportunas.

Según los requisitos principales, la primera de las pantallas está destinada a un esquema general de la aplicación. La segunda de ellas tratará de representar los datos en tiempo real en una serie de gráficas. La última, se usará para consultar datos almacenados en la base de datos, y se mostrarán adecuadamente en gráficas. Pero, conforme avanzaba el proyecto, se especificaron nuevas funcionalidades que dichos programas no podían realizar. Por tanto, se diseñó también otro esquema general, una aplicación que proporciona informes de datos almacenados, un sistema de recogida de ficheros FTP y un interfaz para la interacción entre el usuario con la base de datos.

Si bien LabVIEW cuenta con muchas funciones de muy distinto tipo, en determinadas ocasiones se ha creído conveniente crear una serie de funciones propias, que están detalladas en el apartado 3.2.10. Para cada uno de los programas principales creados, se especifica su descripción, el interfaz con el usuario (Panel frontal), su funcionamiento (Diagrama de bloques) y la jerarquía de funciones utilizadas.

3.2.2. Esquema General

a) Descripción

El esquema general (EsquemaGeneral.vi) detalla el conexionado eléctrico entre los distintos componentes de la microrred. Tiene como objetivo principal mostrar al usuario datos en tiempo real de los distintos dispositivos. Fue especificado un diseño inicial de este esquema, que se puede ver en la Figura 3.9.

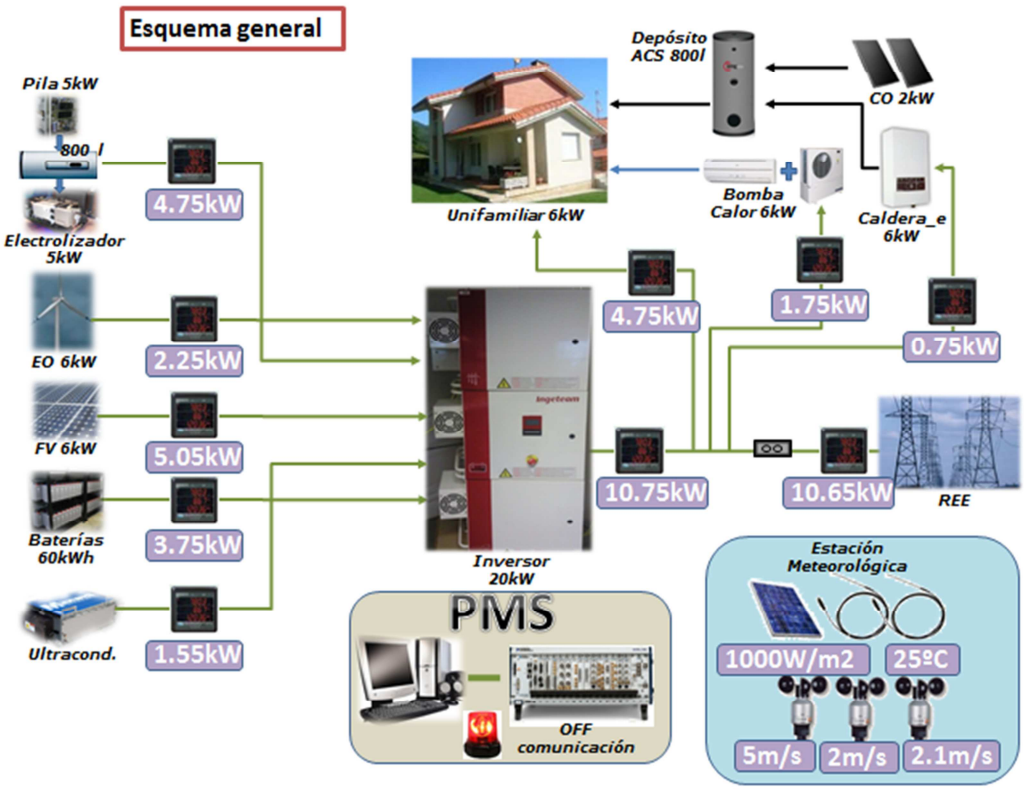

Figura 3.9: Prototipo Esquema general

En el centro se puede apreciar el Inversor Híbrido, la parte central de todo el sistema. A la derecha se concentrarían los datos simulados. También estarían presentes los distintos aparatos pertenecientes a la estación meteorológica, con sus correspondientes valores. A la izquierda del inversor se pueden ver el resto de dispositivos pertenecientes a la microrred, con sus valores actuales. Además, se muestra la comunicación entre el PC y el PXi, y una señal de alarma se activaría si hubiese algún problema respecto al conexionado eléctrico. Esta última restricción no se ha llevado a cabo, si no que sólo se han tenido en cuenta los posibles errores en la comunicación, como se verá más adelante en el Esquema de Comunicación.

La idea de este esquema es poder ver de una manera sencilla los distintos datos de cada uno de los dispositivos, pudiendo elegir en todo momento cuál de los datos se desea visualizar.

b) Panel Frontal

Icono: 

Representa la transmisión de datos al PC y los menús de números dobles que muestran los datos.

Se produjo una versión inicial del Esquema general que se puede ver en la Figura 3.10.

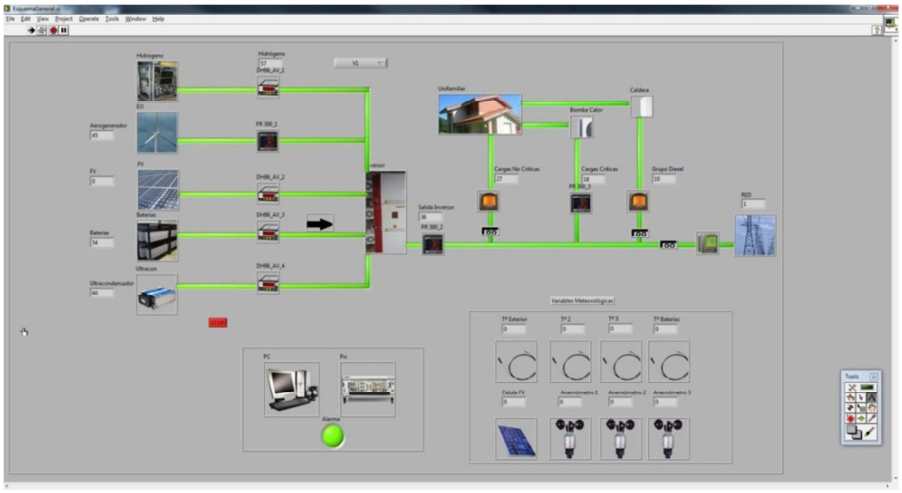

Figura 3.10: Esquema general (versión 1)

Sin embargo, se consideró que se desaprovechaba mucho espacio del disponible y se modificó repetidamente hasta obtener el mayor rendimiento del tamaño de la pantalla. Además, se eliminó la parte que contenía la caldera y el depósito, arriba a la derecha. Esto dio lugar a la creación de un Esquema General Térmico. Así pues, el actual no contiene esa información, y se ha reorganizado y reajustado el tamaño y la disposición de todos los elementos. El resultado se puede ver en la Figura 3.11.

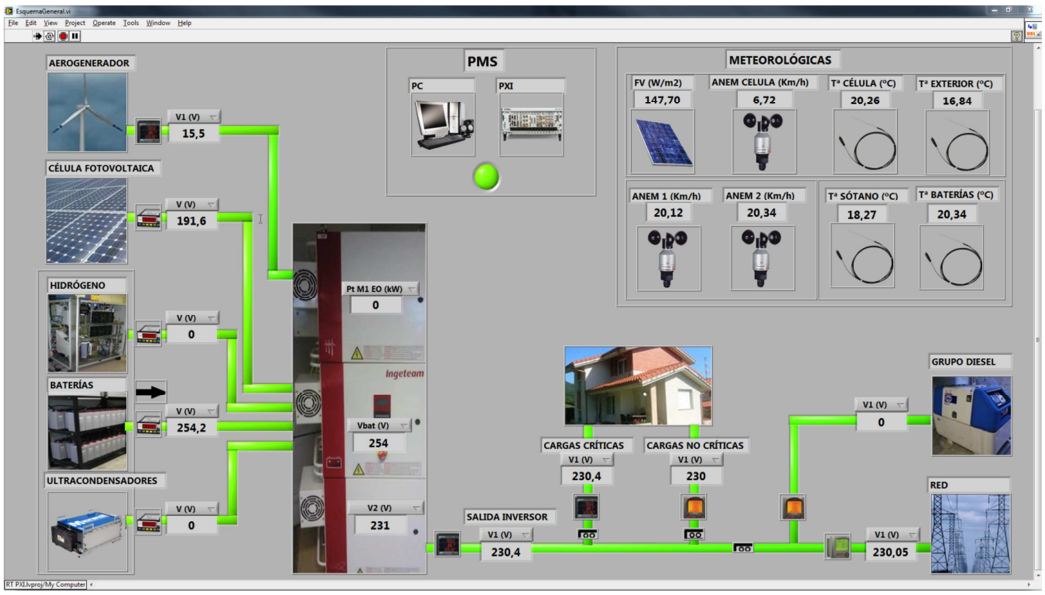

Figura 3.11: Esquema general (versión final)

El funcionamiento del Esquema General es muy sencillo: cada dispositivo cuenta con un menú y un indicador. Según la opción elegida en el menú, se mostrará la información correspondiente en el indicador. El menú contiene los distintos valores que pueden ser representados de un dispositivo. Además, se muestra el tipo de analizador de red al que está conectado, pudiendo ser CVM k2, CVM NRG96, DH96 AV o PR300. Así pues, para el Aerogenerador, vemos en la Figura 3.12 las posibilidades del menú, y en las Figuras 3.13 y 3.14 valores correspondientes a la opción elegida.

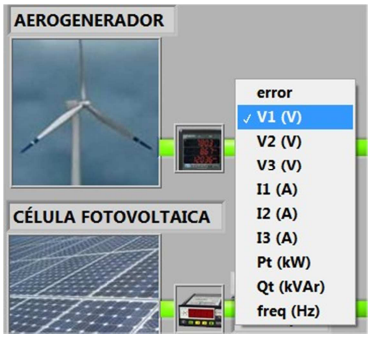

Figura 3.12: Menú

Figura 3.13: V1

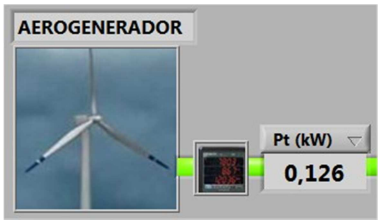

Figura 3.14: Pt

c) Diagrama de bloques

El diagrama de bloques del Esquema General, como en casi todos los Vis, consta de dos partes diferenciadas: una de configuración (Figura 3.15 y Figura 3.16) y otra de ejecución (Figura 3.17).

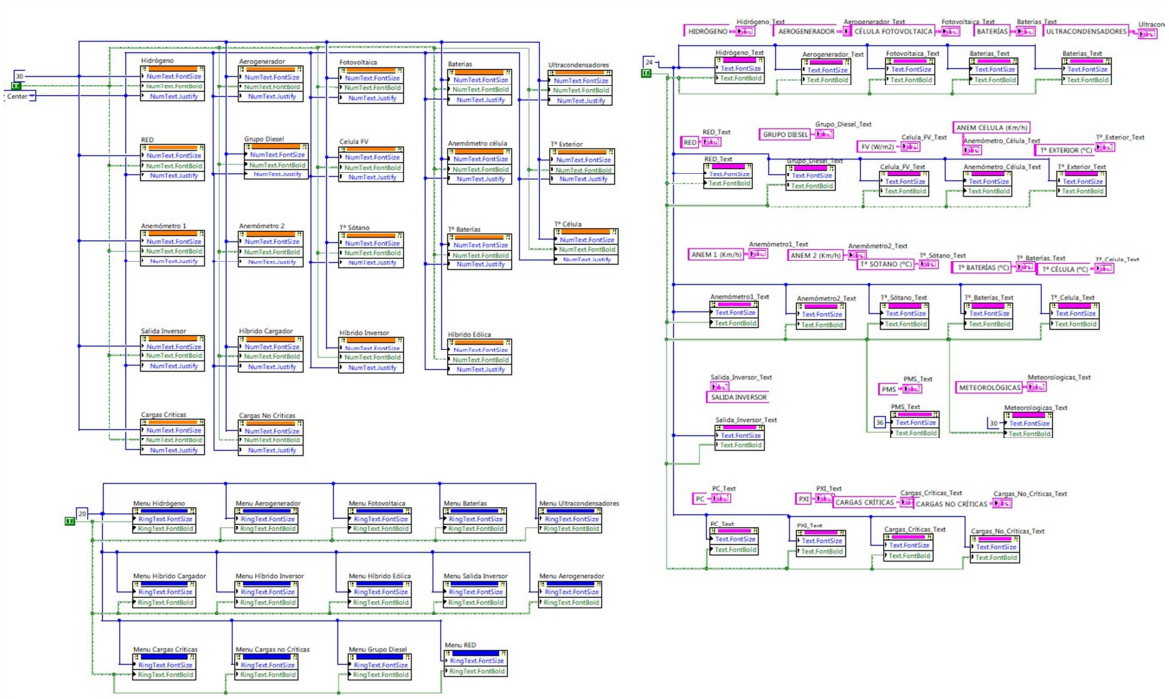
Figura 3.15: Esquema General: configuración

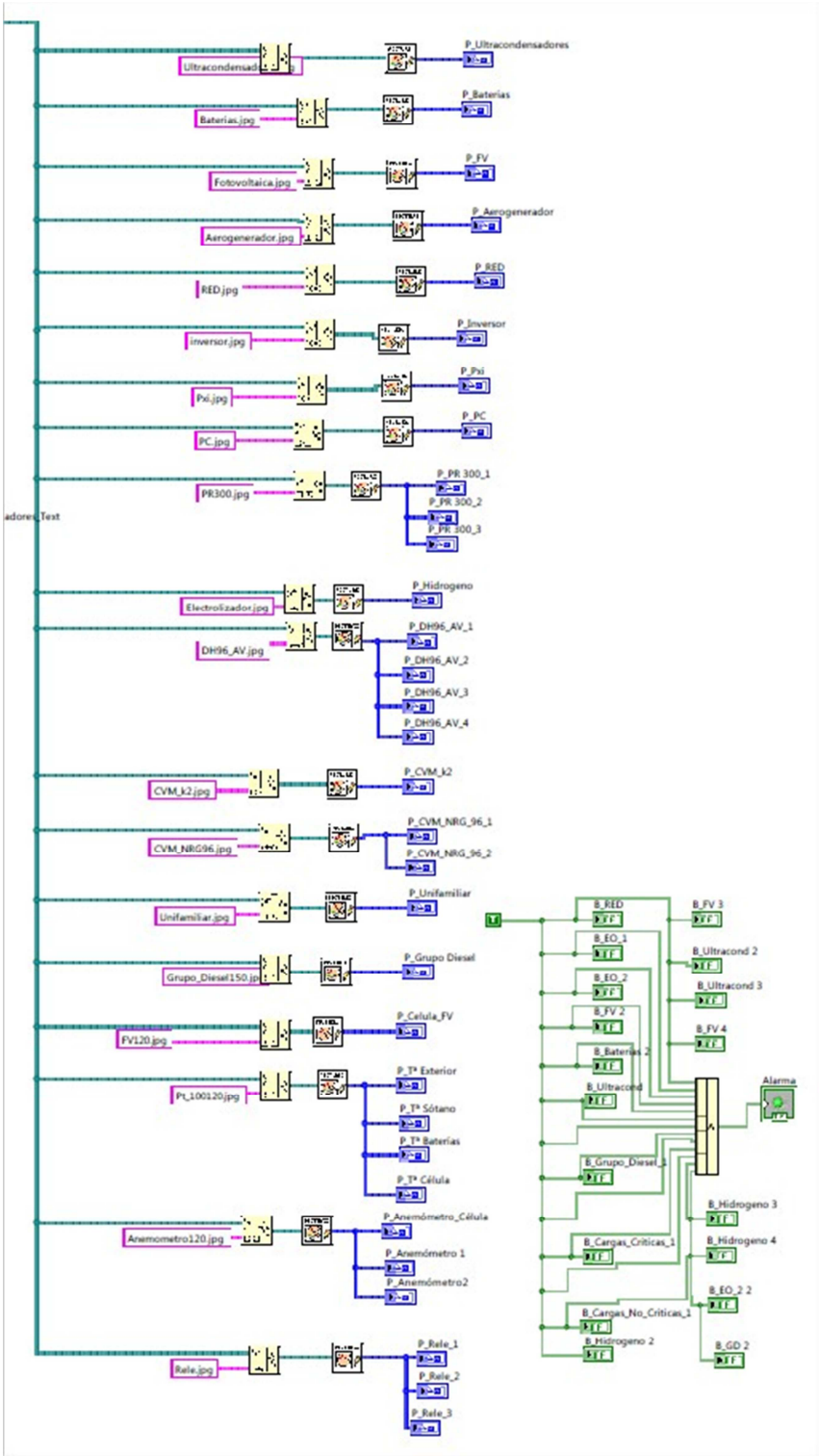

Figura 3.16: Esquema General: configuración

En la parte de configuración se especifican todo lo relacionado al aspecto puramente visual: tamaño de letras, imágenes,…

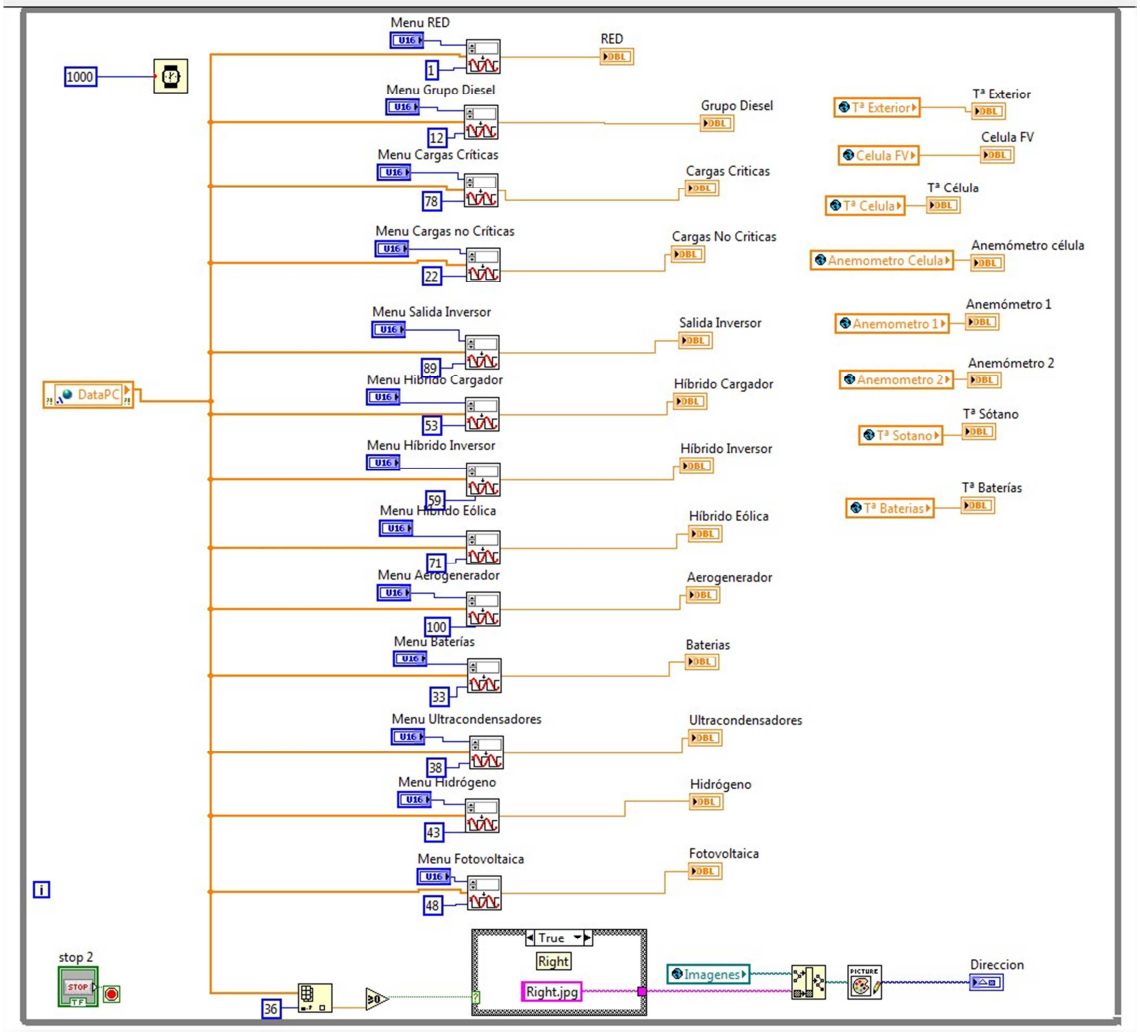

Figura 3.17: Esquema General: ejecución

En la Figura 3.18, se define para el indicador del Hidrógeno el tipo de número. En este caso, tamaño 30, en negrita y centrado.

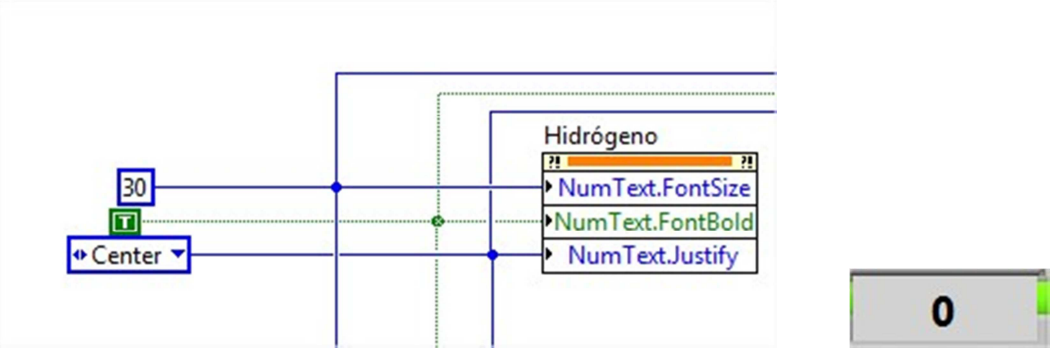

Figura 3.18: Especificación de indicador en Hidrógeno y resultado

En la Figura 3.19 se define para el cuadro de Menú del Hidrógeno, el tamaño 20 para la letra, y el tipo negrita.

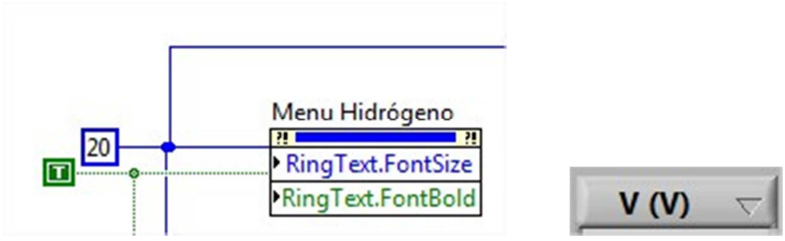

Figura 3.19: Especificación del Menú en Hidrógeno y resultado

En la Figura 3.20 se define para el título del Hidrógeno, un tamaño de letra 24 y en negrita.

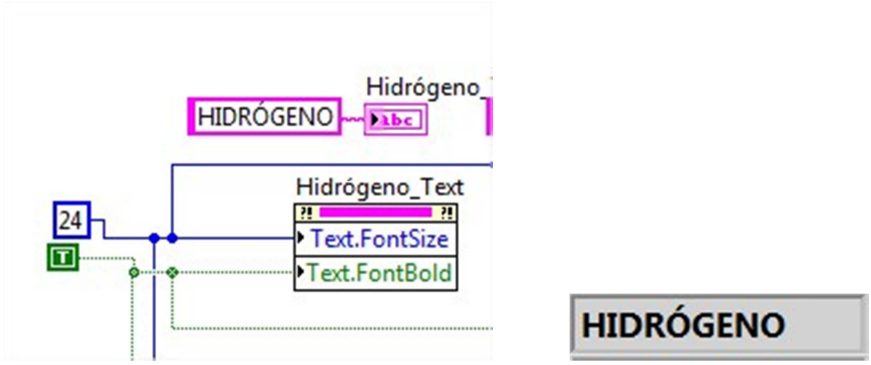

Figura 3.20: Título Hidrógeno y resultado

En la Figura 3.21 se define para el cuadro de imagen de Baterías la ruta de dicha imagen, en jpeg. La ruta parcial está definida en una variable global, que se usarán en todo el proyecto.

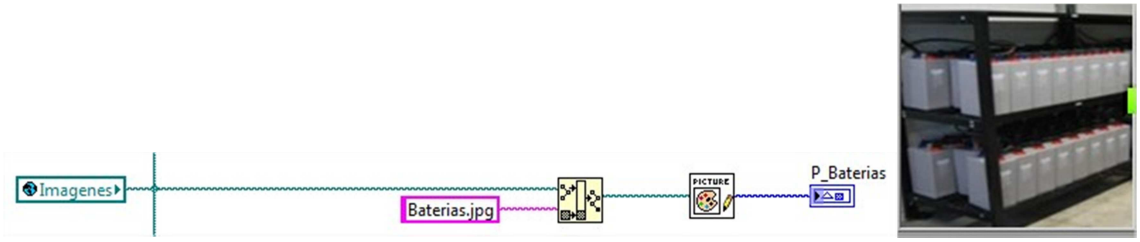

Figura 3.21: Imagen Baterías y resultado

Las especificaciones de imágenes, títulos e indicadores son similares a lo largo de todo el proyecto, de modo que no se volverán a definir.

La parte de ejecución del Esquema General se basa en un bucle infinito, que se ejecuta cada segundo, ya que es la frecuencia con la que se generan datos, como se puede ver en la Figura 3.22. Este bucle basa su funcionamiento en leer el array de datos de entrada, especificado como DATAPC. Según el dispositivo y la opción especificada en el Menú, el subVI MenuToChart.vi devolverá el dato correcto del array, que se representará en el indicador adecuado. También son representados todos los datos meteorológicos, cuyos valores proceden de unas variables globales a las que se les asigna valor desde el VI de gráficas en tiempo real. Por último, la dirección de la corriente en las Baterías dependerá del valor de la intensidad (I), en nuestro caso, la posición 36 del array. Así, según el valor de dicho dato, representaremos una flecha u otra.

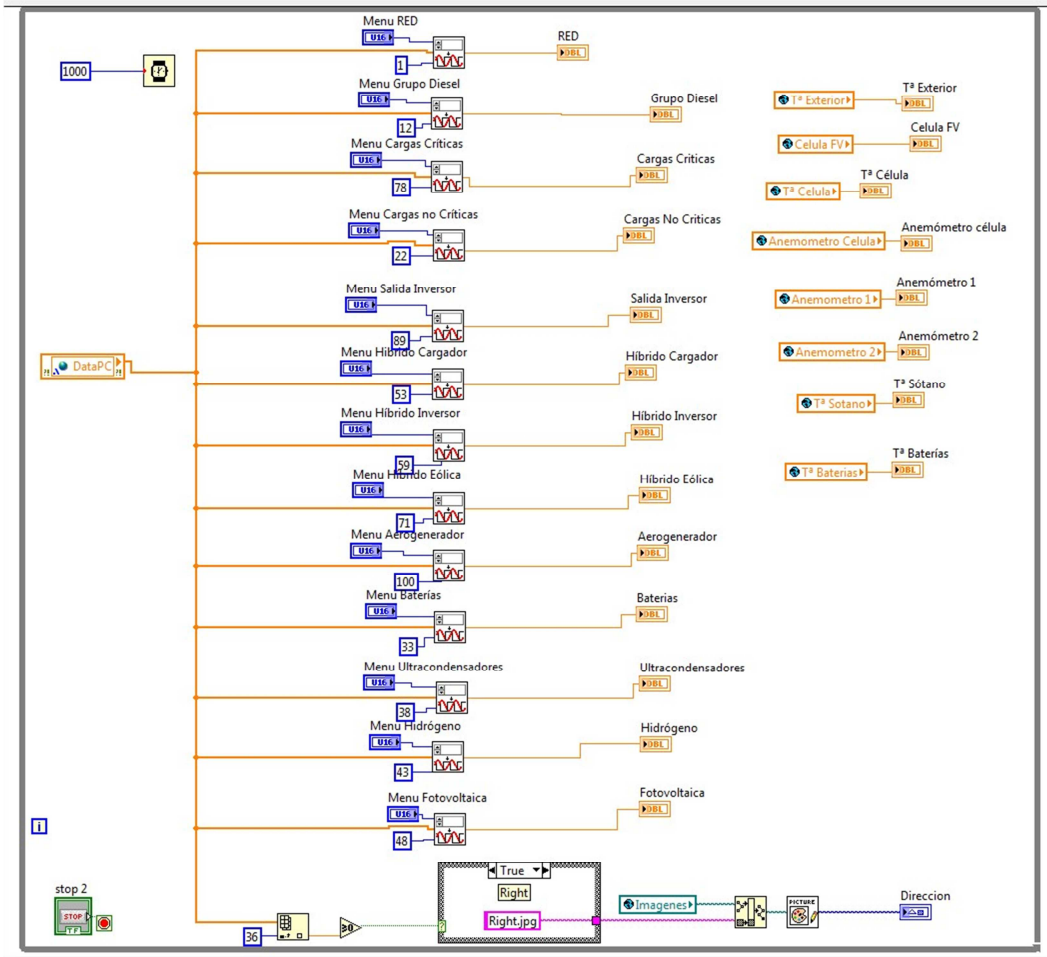

Figura 3.22: Diagrama de bloques del Esquema General

d) Jerarquía del VI

Los SubVIs se pueden ver en la Figura 3.23. Para consultar su funcionamiento interno, dirigirse a la sección 3.2.10.

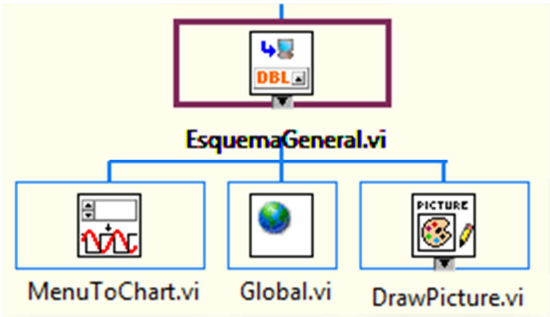

Figura 3.23: SubVIs utilizados

SubVIs utilizados:

- MenuToChart
- Global
- DrawPicture

3.2.3. Esquema General Térmico

a) Descripción

El Esquema General Térmico (EsquemaGeneralTermico.vi) se puede ver como otra versión del Esquema General, ya que también indica el conexionado eléctrico entre los distintos dispositivos y la información en tiempo real de cada indicador. Sin embargo, el Esquema General Térmico provee de la información perteneciente a la parte térmica de la microrred, cosa que el Esquema General no hace. Ambos Esquemas Generales son similares, por lo tanto, en ningún momento se ejecutarán simultáneamente, si no que se elegirá uno u otro en función de los datos de interés en ese momento.

En el Esquema General Térmico, con diferencia del anterior, se ha eliminado la representación de la conexión entre PXi y PC. Además, las variables meteorológicas han quedado en un segundo plano, con objetivo de dar cabida a toda la parte térmica, que toma importancia. La parte térmica está compuesta por una bomba de calor, una caldera, una placa solar y un depósito, que están conectados con el unifamiliar. Todos estos datos son simulados, y por cada uno de los dispositivos sólo tenemos un dato, por lo tanto, como sucede en las variables meteorológicas, no habrá menú para seleccionar la variable deseada, ya que resulta trivial al haber solo una.

b) Panel Frontal

Icono: 

Representa la transmisión de datos al PC y los menús de números dobles que muestran los datos.

Podemos observar en la Figura 3.24 que el interfaz con el usuario del Esquema General Térmico es muy similar al Esquema General, con los cambios comentados. Su funcionamiento también es similar a ese esquema, ya que únicamente se han añadido algunos dispositivos.

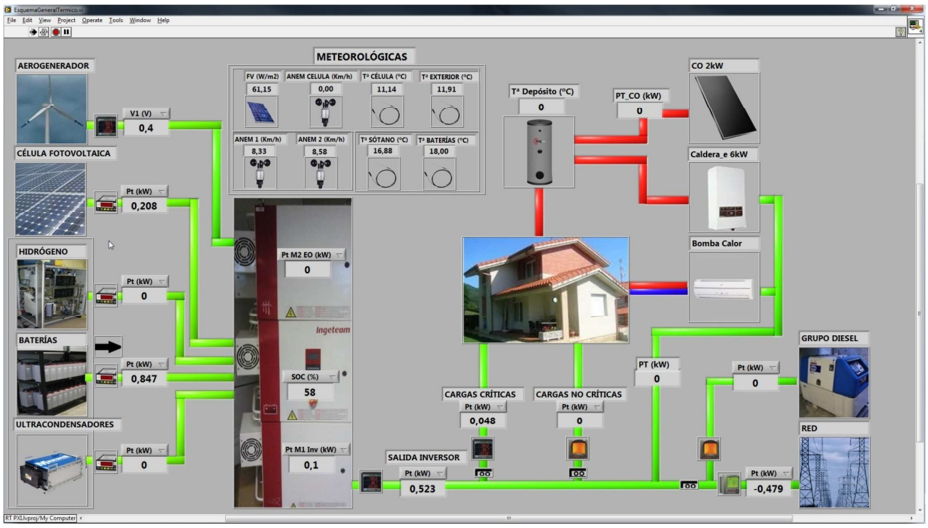

Figura 3.24: Esquema General Térmico

c) Diagrama de bloques

El diagrama de bloques del Esquema General Térmico es similar al del Esquema General. La única parte distinta respecto a la ejecución se aprecia en la Figura 3.25, en la que se obtienen los valores de las variables nuevas.

Figura 3.25: Cambios en el Esquema General Térmico

d) Jerarquía del VI

Los SubVIs se pueden ver en la Figura 3.26. Para consultar su funcionamiento interno, dirigirse a la sección 3.2.10.

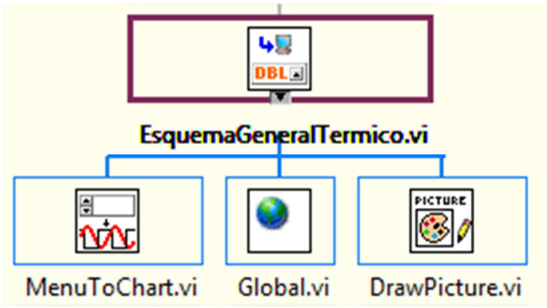

Figura 3.26: SubVIs utilizados

SubVIs utilizados:

- MenuToChart
- Global
- DrawPicture

3.2.4. Esquema de Comunicación

a) Descripción

El Esquema de Comunicación (EsquemaComunicacion.vi) detalla el conexionado de comunicación entre los distintos componentes de la microrred. Informa del tipo de comunicación entre los dispositivos según un código de colores que se puede consultar en una leyenda en el mismo vi. Además, en caso de error de comunicación, dicho conexionado se mostrará en rojo y se encenderá una alarma general. La otra funcionalidad del Esquema de Comunicación, es insertar en la base de datos los datos correspondientes en cada momento. Fue especificado un diseño inicial de este esquema, que se puede ver en la Figura 3.27.

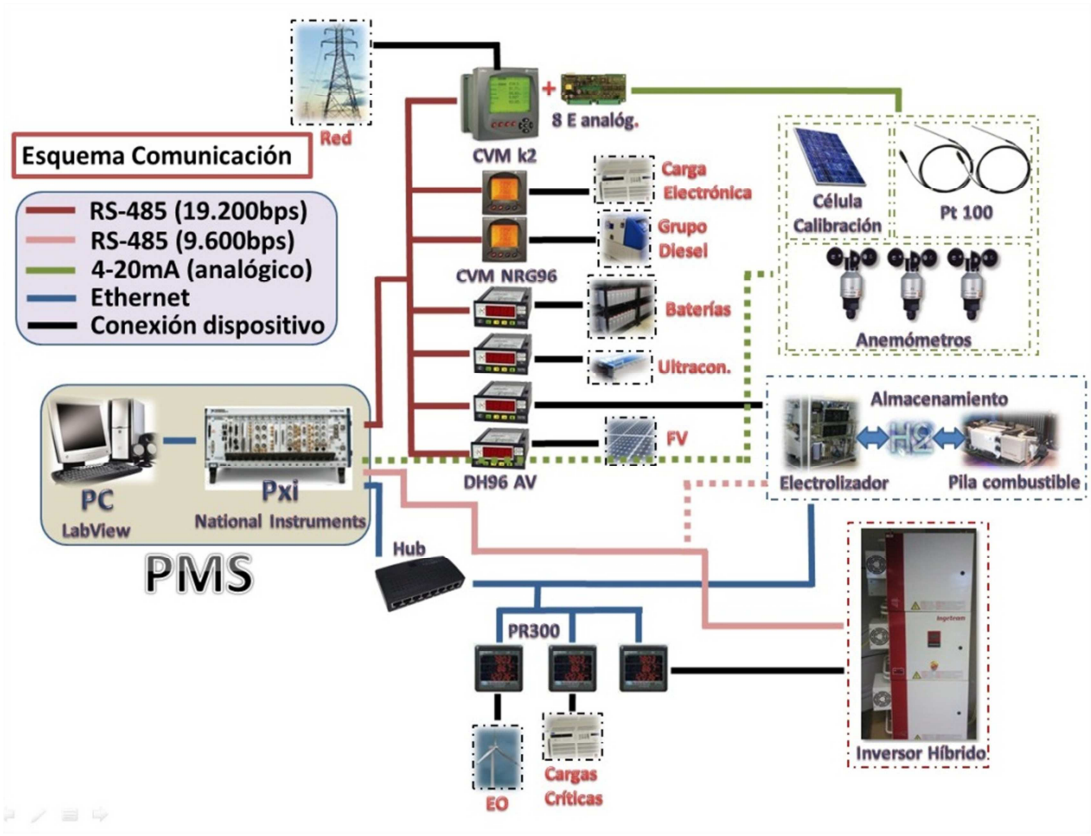

Figura 3.27: Prototipo Esquema de Comunicación

En el Esquema de Comunicación podemos ver el tipo de conexión entre el PXi y el resto de dispositivos, por lo tanto, el PXi aparece a la izquierda, siendo el elemento central, y de él parten todas las conexiones. Por un lado, Ethernet con el PC, y por otro, de distintos tipos para el resto de dispositivos. Dichos dispositivos están agrupados según tipo de comunicación. Este tipo de comunicación puede ser RS-485(19.200 bps), RS-485(9.600bps), 4-20mA (analógico), Ethernet y conexión con el dispositivo). Los cuatro primeros tipos se dan desde el PXi hasta los indicadores, mientras que la conexión con el dispositivo se produce entre los indicadores y los dispositivos propiamente dichos.

La idea de este Esquema de Comunicación consiste en apreciar fácil y rápidamente los posibles errores de comunicación de la microrred, de una manera visual y amigable.

b) Panel Frontal

Icono: 

Representa la inserción de los datos en la base de datos

Se produjo una versión inicial del Esquema general que se puede ver en la Figura 3.28.

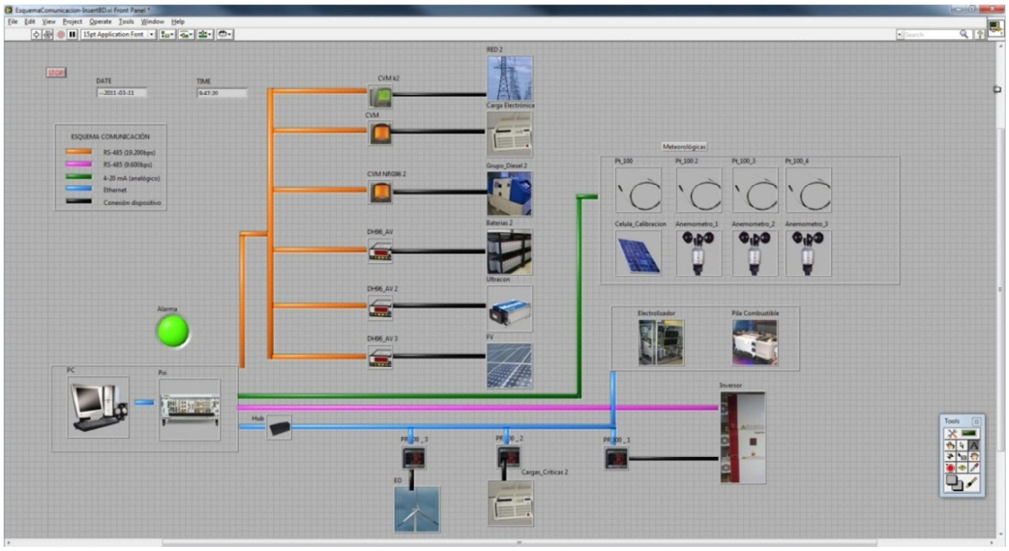

Figura 3.28: Esquema de Comunicación (versión 1)

Al igual que en el Esquema General, en el Esquema de Comunicación se consideró que se desaprovechaba mucho espacio del disponible y se modificó para ocupar toda la pantalla, de modo que fuese más sencillo de visualizar. Además, se estableció una nueva conexión entre el PXi y los dispositivos del Hidrógeno (Electrolizador y Pila combustible), de tipo RS-485 (19.200bps). El resultado de estas modificaciones se puede ver en la Figura 3.29.

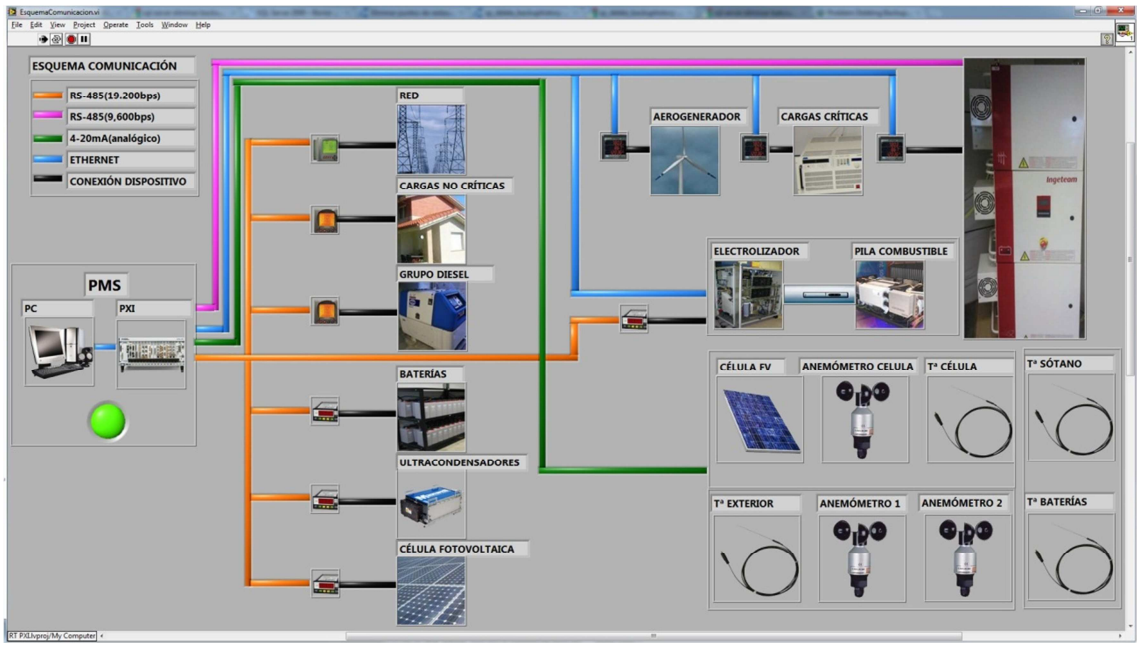

Figura 3.29: Esquema de Comunicación (versión final)

El Esquema de Comunicación no requiere intervención por parte del usuario para su funcionamiento, si no que se ejecuta automáticamente sin posibilidad de interacción. Al recibir los datos, considerará si se ha producido algún error de comunicación, y, en ese caso, alertará cambiando el color de las conexiones, y encendiendo la alarma general, situada bajo el PC y el PXi. La inserción de los datos en la base de datos, también se produce automáticamente y sin ningún tipo de intervención. Por lo tanto, se puede afirmar que este VI no es tanto de interacción, si no de consulta.

c) Diagrama de bloques

El diagrama de bloques del Esquema de Comunicación, como en casi todos los Vis, consta de dos partes diferenciadas: una de configuración (Figura 3.30) y otra de ejecución (Figura 3.31). La parte de configuración es obviada al ser prácticamente igual que lo especificado en el Esquema General (3.2.2).

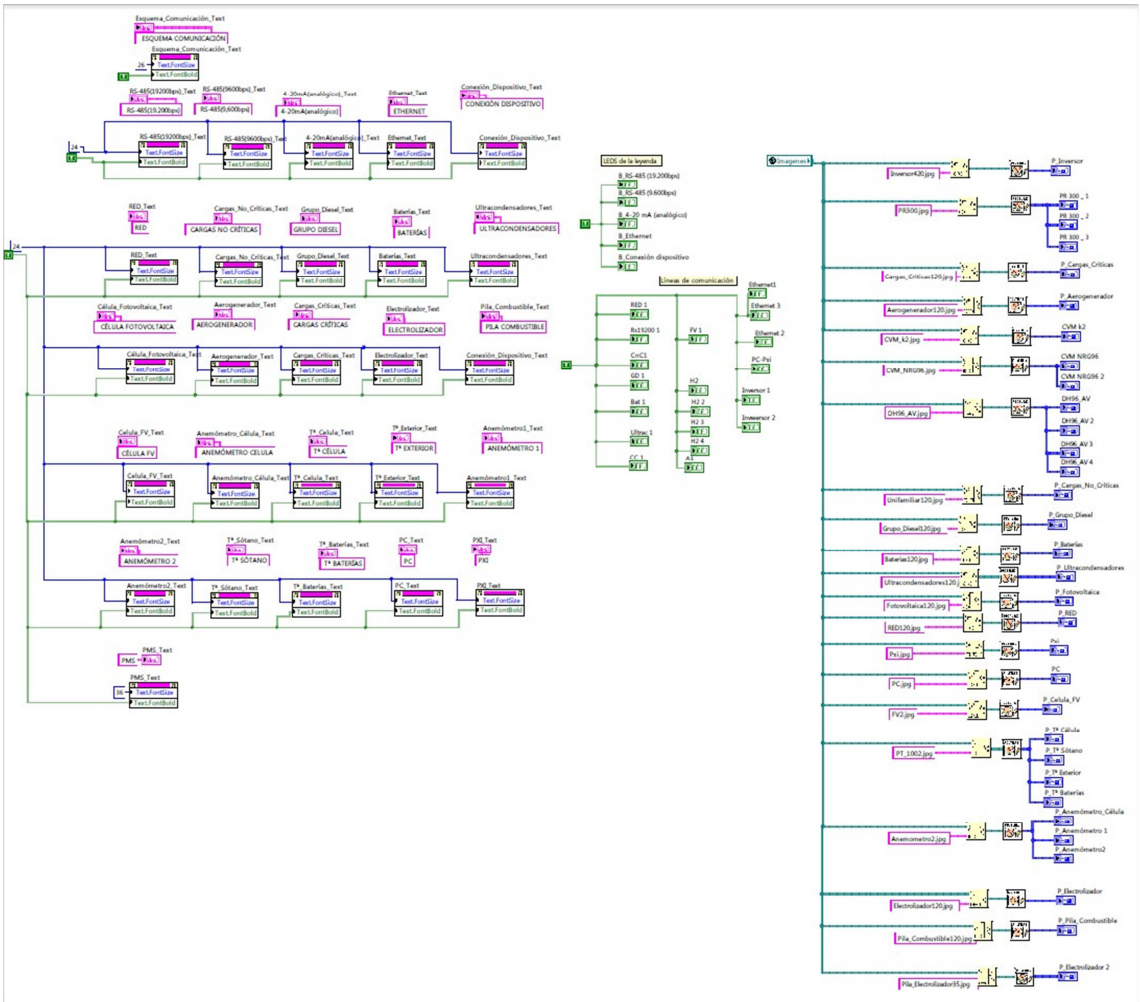

Figura 3.30: Esquema de Comunicación: configuración

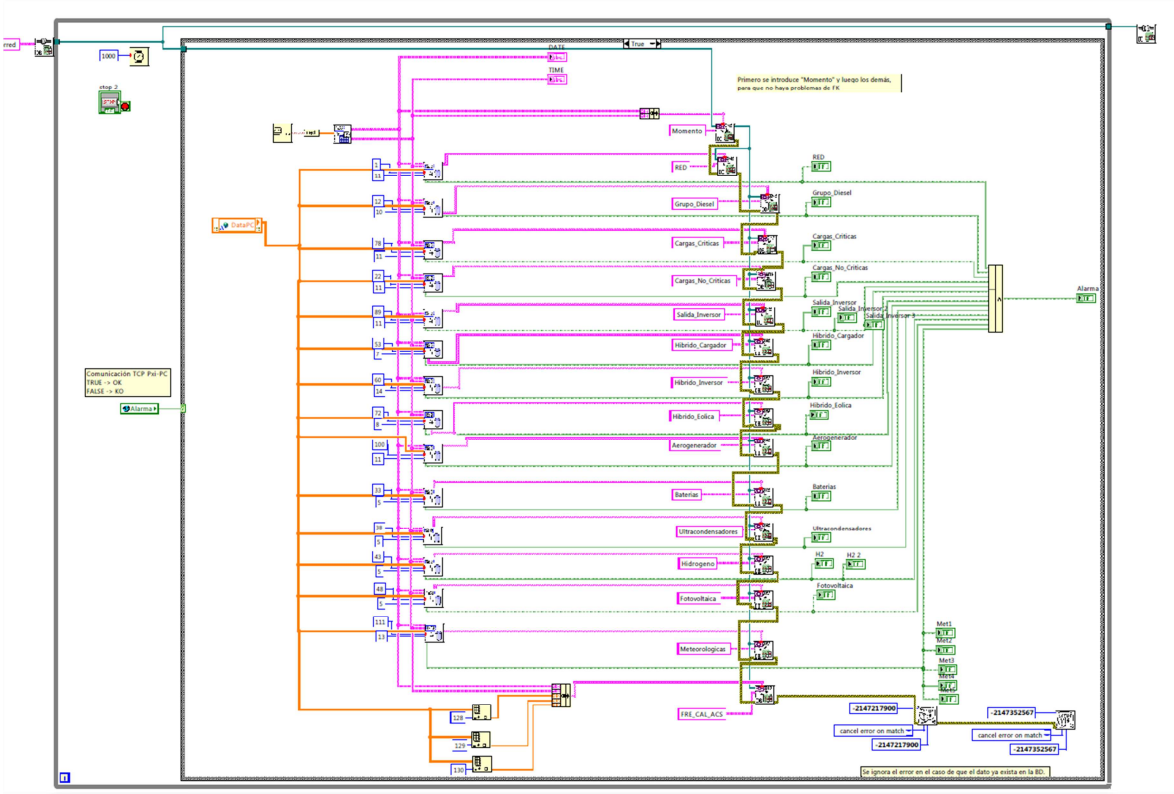

Figura 3.31: Esquema de Comunicación: ejecución

La parte de ejecución se ejecuta dentro de un bucle infinito. Antes de dicho bucle, se establece la conexión con la base de datos, de modo que con una única conexión insertemos todos los datos. Esto se ha hecho así ya que LabVIEW tarda en establecer la conexión con la base de datos unos segundos, y si se establecía y cerraba la conexión para cada grupo de datos de 1 segundo, era imposible recogerlos todos. Además, resulta mucho más eficiente abrir una única conexión, que será cerrada en caso de finalización del bucle. LabVIEW no permite bucles infinitos, si no que realmente son bucles que pararán al presionar un botón llamado STOP. Sin embargo, si ocultamos dicho botón al usuario (en el diagrama de bloques, clic derecho y "Hide Control") nunca parará. Esto significa también que no se cerrará la conexión de la base de datos, ya que mientras no se pulse la aplicación continuará insertando datos.

Como se especificó en el apartado de OBDC (31.1.3), se ha creado un DNS con el nombre "Microrred" para realizar la conexión con la base de datos. Una vez creado, para establecer la conexión desde LabVIEW, solo se necesita una de las funciones ya proporcionadas por el software, y añadir ese literal, como podemos ver en la Figura 3.32.

Figura 3.32: Conexión con la base de datos

Teniendo en cuenta que la conexión entre el Pxi y el PC puede fallar, por lo que no llegarán los datos adecuados, se ha creado una variable global llamada Alarma que valdrá TRUE siempre que la conexión esté correctamente establecida. Este valor se obtiene de TCP Communication - Host.vi, programa que no pertenece a mi proyecto. Siempre que valga TRUE se podrán extraer los datos de ese segundo para insertarlo, y si vale FALSE no hará nada.

A la hora de programar este VI, hay que tener en cuenta la estructura de la base de datos. Se comentó que hay una tabla llamada "Momento" que contiene las horas y fechas en las que se producen datos, y que todos los demás dispositivos están relacionados con esa tabla. Así pues, es necesario que una hora y fecha exista en "Momento" antes de ser insertados los datos en los demás dispositivos. Esto se ha conseguido llevando la conexión de la base de datos a la función que inserta los datos en "Momento". La conexión de salida de esa función se lleva a todos los demás módulos. Así, nos aseguramos que, al insertar datos en los módulos, esa hora y fecha ya existe en la tabla "Momento", por lo que no se producirán errores de FK (clave foránea).

La hora y fecha a insertar se obtiene a partir de la primera posición del array recibido. Ese dato, que llega como doble, se trata en la función creada DoubleToDateTime, que devuelve dos cadenas de caracteres, una con la fecha y otra con la hora. (Figura 3.33).

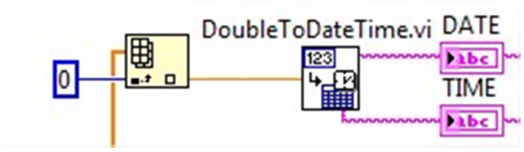

Figura 3.33: Conversión a fecha y hora

Los datos numéricos a insertar en cada módulo de la base de datos se obtienen a partir del array de datos, la hora, la fecha y la función creada ArrayToDB. Estos datos devueltos, son llevados a la función propia de LabVIEW DBToolsInsertData, a la que únicamente le pasamos como parámetro los datos y la tabla en la base de datos en la que se deben insertar los mismos. (Figura 3.34)

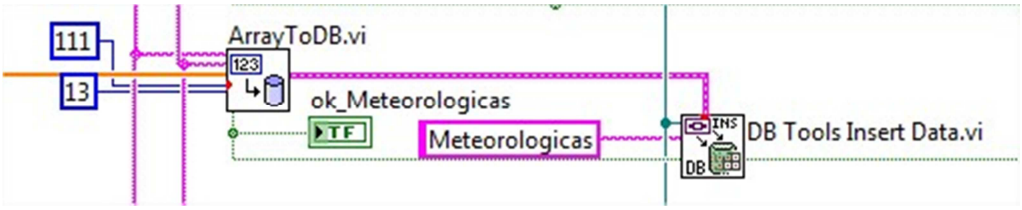

Figura 3.34: Extracción de datos e inserción en la base de datos

La función ArrayToDB, además de obtener los datos, devuelve un booleano que indicará si el dato de error de ese dispositivo indica algún tipo de error (timeout,…) o no (vale 0). Este booleano devolverá TRUE si no se ha producido ningún error de comunicación. Así pues, para establecer la alarma general, se juntan todos los valores booleanos que puedan indicar error con una función AND, que devolverá TRUE si todos los booleanos pasados también lo valían.

Por último, comentar que para el dispositivo FRE+CAL+ACS no se ha utilizado la función creada ArrayToDB ya que fueron añadidos en el último momento, había posibilidades de cambio, y como son simulados no siguen el mismo formato que los demás, ya que no contienen en ningún lugar el número del módulo, variable de error…

Por ello, simplemente se han extraído los datos de las tres posiciones ocupadas, se han unido con la fecha y la hora, y se han insertado, de modo que un cambio en el número de datos simulados, etc, sería muy sencillo de solucionar.

Por último, se ha añadido un código que ignora los errores con códigos -2147217900 y -2147352567 en caso de que se produzcan, que corresponden a errores que tienen que ver con violaciones de las claves primarias (los datos ya existen) o intento de inserción de datos erróneos (la fecha en un formato desconocido, etc). Si bien estos errores no deberían darse nunca, se ha decidido añadir código para ignorarlos ya que un fallo en el vi provoca que todo el proyecto se detenga, y no se desea eso. Así, si en un momento dado llegan datos erróneos, simplemente se ignorarán.

d) Jerarquía del VI

Los SubVIs se pueden ver en la Figura 3.35. Para consultar su funcionamiento interno, dirigirse a la sección 3.2.10.

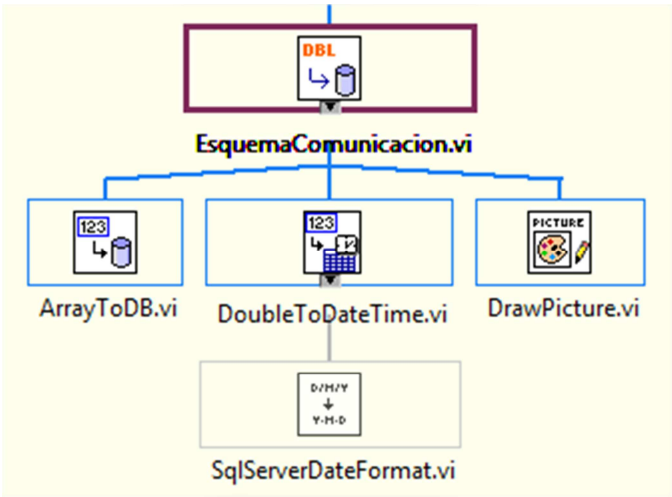

Figura 3.35: SubVIs utilizados

SubVIs utilizados:

- ArrayToDB
- DoubleToDateTime
  - SqlServerDateFormat
- DrawPicture

3.2.5. Gráficas en tiempo real

a) Descripción

Las gráficas en tiempo real (RealTimeCharts.vi) constan de una serie de gráficas que representan en tiempo real los datos producidos por los dispositivos. La idea era poder visualizar claramente, y sin tener que acceder a la base de datos, los valores más recientes de los distintos campos de los dispositivos, para ver su evolución y comprobar que funcionan correctamente, y que se dan los valores esperados. Fue especificado un diseño inicial de este esquema, que se puede ver en la Figura 3.36.

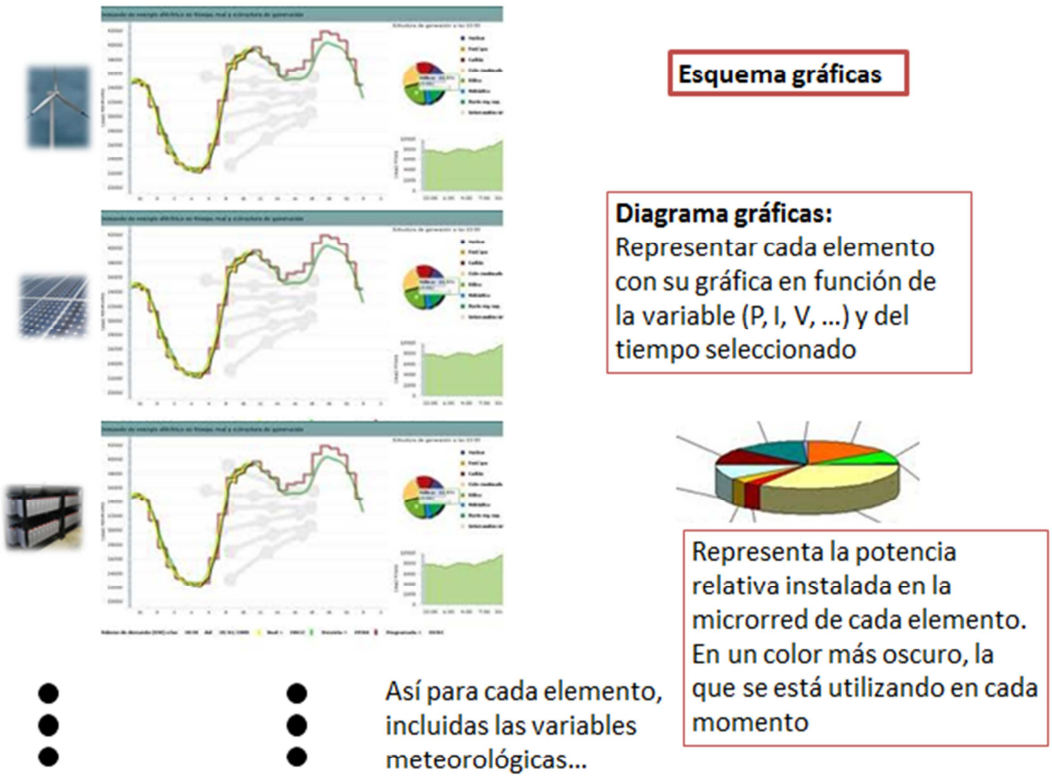

Figura 3.36: Prototipo gráficas en tiempo real

Para cada dispositivo habría una gráfica en la que se elegiría cuál de los datos disponibles representar: potencia, intensidad,… También habría una gráfica en forma de tarta que representaría la potencia utilizada por cada elemento del total. Este último requisito no se ha podido llevar a cabo debido a las grándes limitaciones de LabVIEW con este tipo de gráficas.

La idea de este VI era poder visualizar al instante la evolución de los distintos dispositivos, con el objetivo de descubrir posibles errores en su funcionamiento o para contrastar los datos devueltos con los esperados.

b) Panel Frontal

Icono: 

Representa la conversión desde números dobles a gráficas, en tiempo real.

Se produjo una versión inicial de las gráficas en tiempo real que se puede ver en la Figura 3.37.

Figura 3.37: Gráficas en tiempo real (versión 1)

Sin embargo, poco después aumentó el número de dispositivos del que había que representar datos, y partiendo de este esquema, no resultaba viable reducir las gráficas para que cupieran más en una sola pantalla, ya que eran demasiado pequeñas para poder visualizar los datos con comodidad. Se decidió utilizar un sistema de pestañas, en la que en cada pestaña estuvieran agrupadas las gráficas de los dispositivos relacionados entre sí. De este modo, en la pantalla se mostrarían un máximo de cuatro gráficas, y cada una de ellas tendría un tamaño suficiente y adecuado.

Las seis pestañas con sus respectivas gráficas son:

- Almacenamiento (Figura 3.38)
  - Baterías
  - Hidrógeno
  - Ultracondensadores

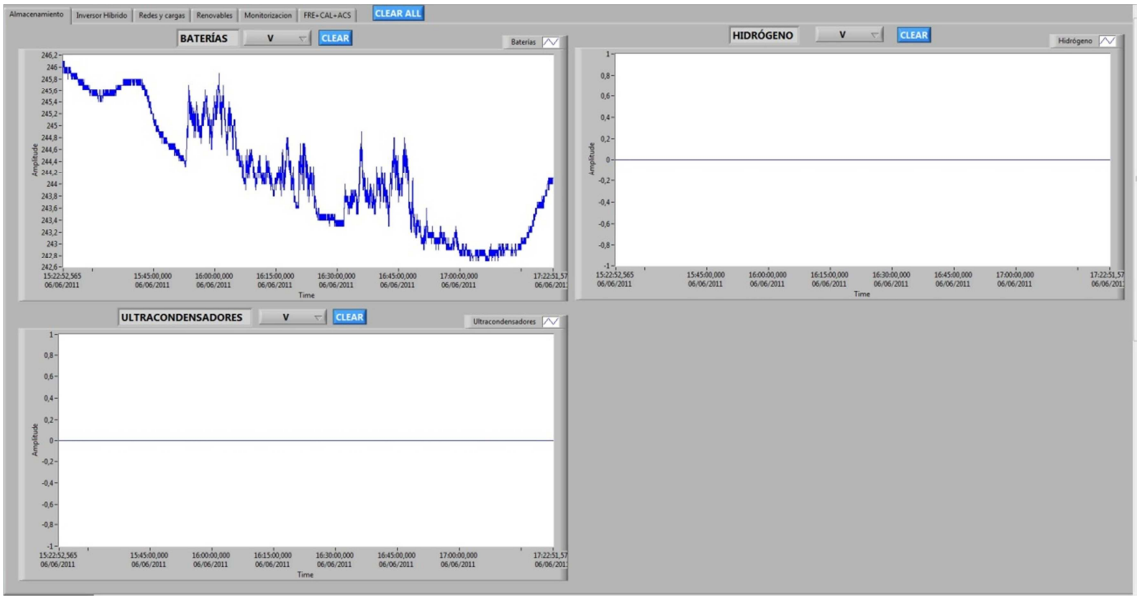

Figura 3.38: Almacenamiento

- Inversor Híbrido (Figura 3.39)
    - Salida Inversor
    - Híbrido Cargador
    - Híbrido Inversor
    - Híbrido Eólica

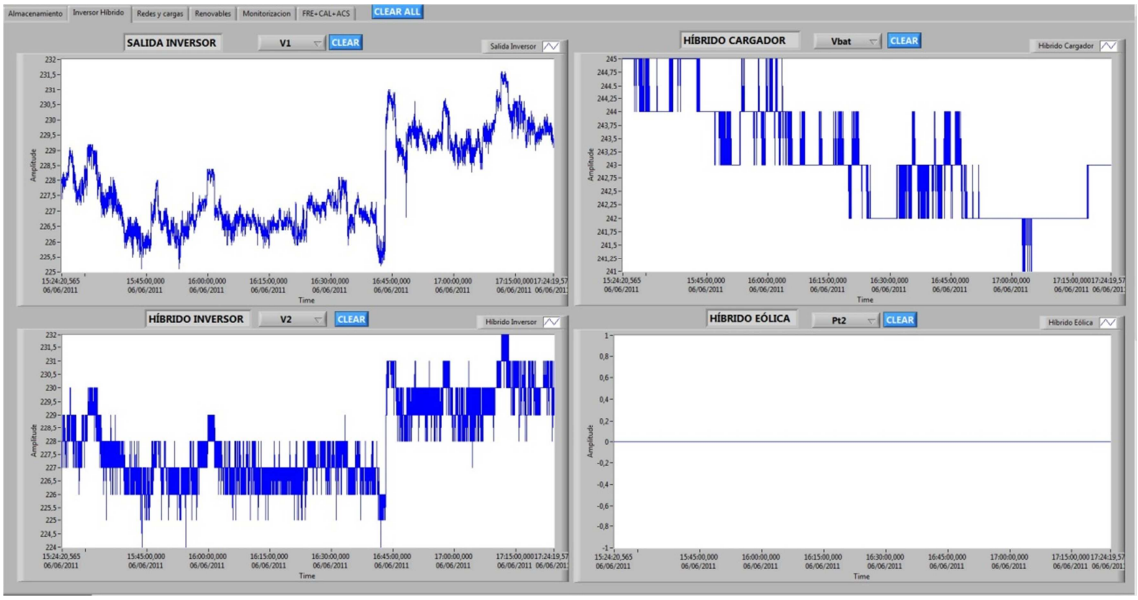

Figura 3.39: Inversor Híbrido

- Redes y cargas (Figura 3.40)
    - RED
    - Grupo Diesel
    - Cargas Críticas
    - Cargas No Críticas

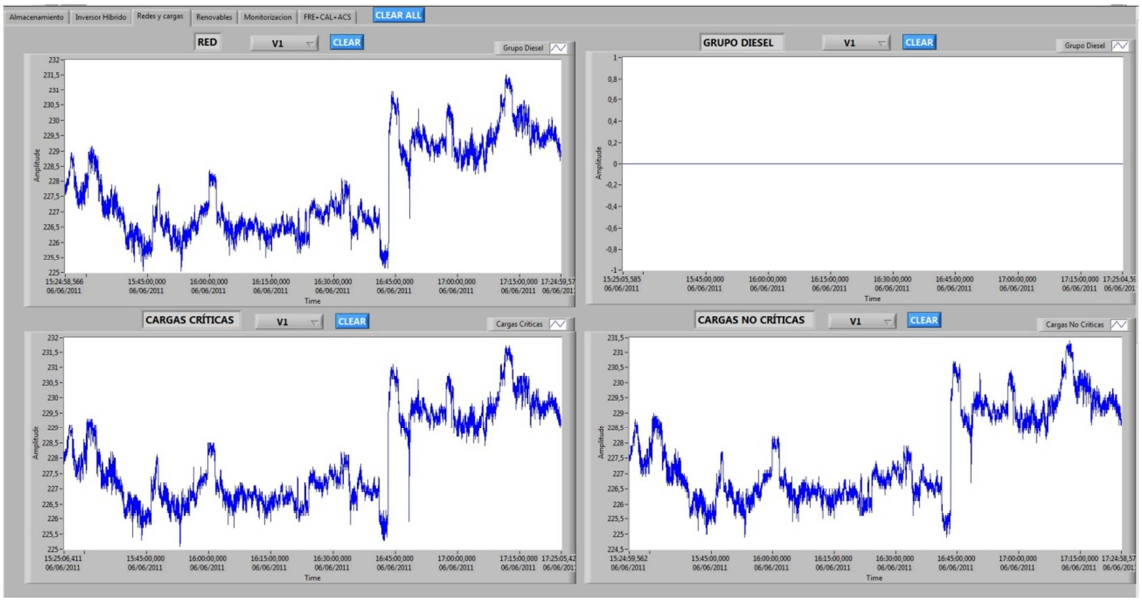

Figura 3.40: Redes y cargas

- Renovables (Figura 3.41)
    - Fotovoltaica
    - Meteorológicas
    - Aerogenerador

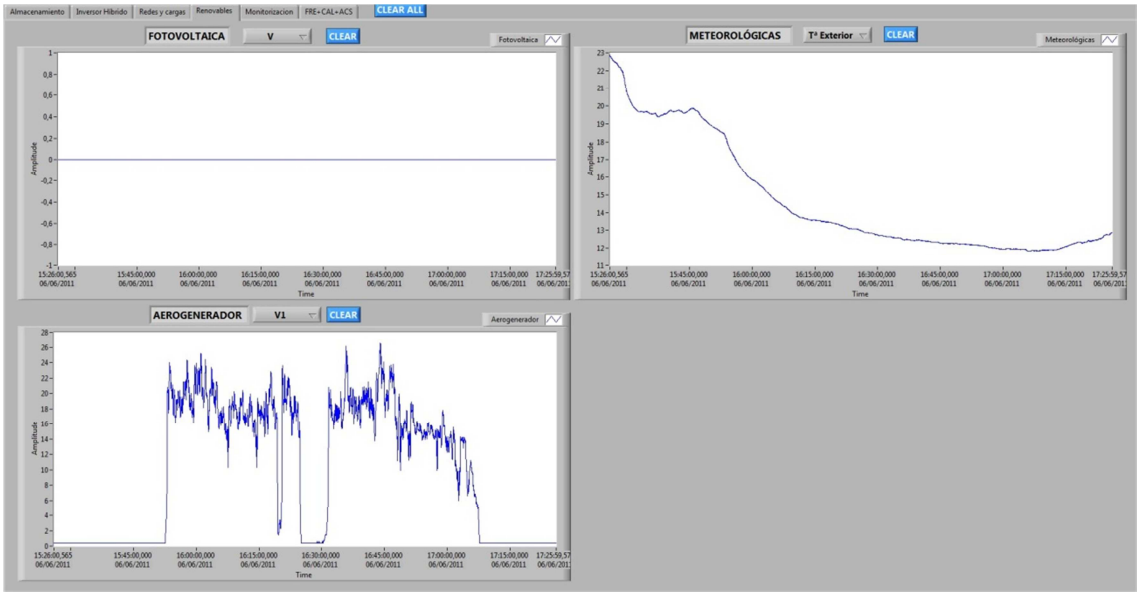

Figura 3.41: Renovables

- Monitorización (Figura 3.42)
    - Potencia FV + EO
    - Potencias de Red
    - Potencia consumida
    - Potencia baterías

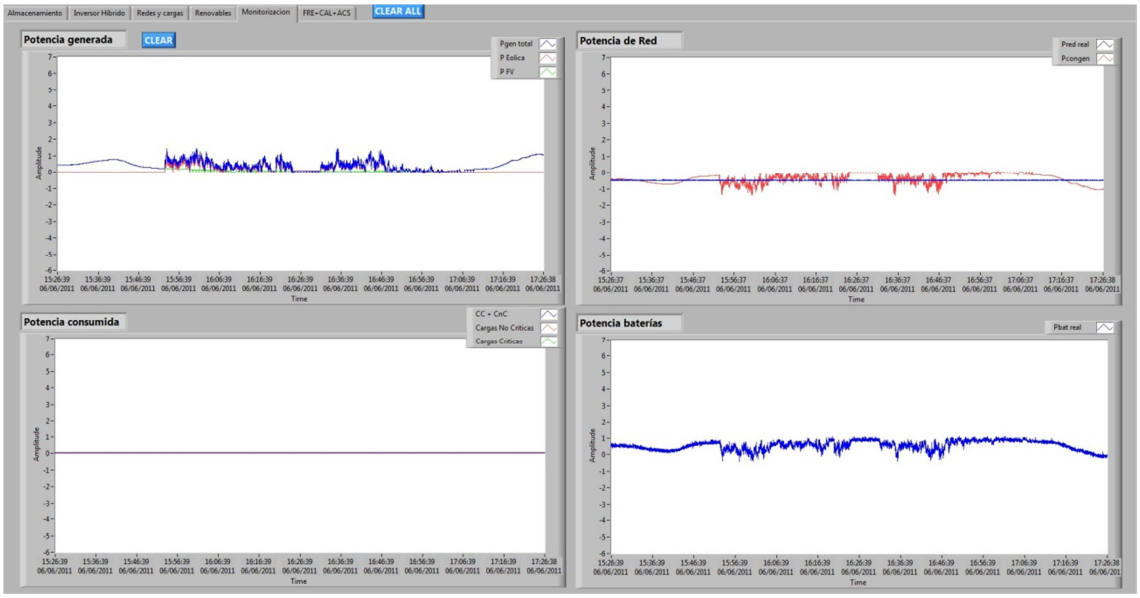

Figura 3.42: Monitorización

- FRE + CAL + ACS (Figura 3.43)
    - Tª Depósito
    - Pt CO 2kW
    - Pt Caldera

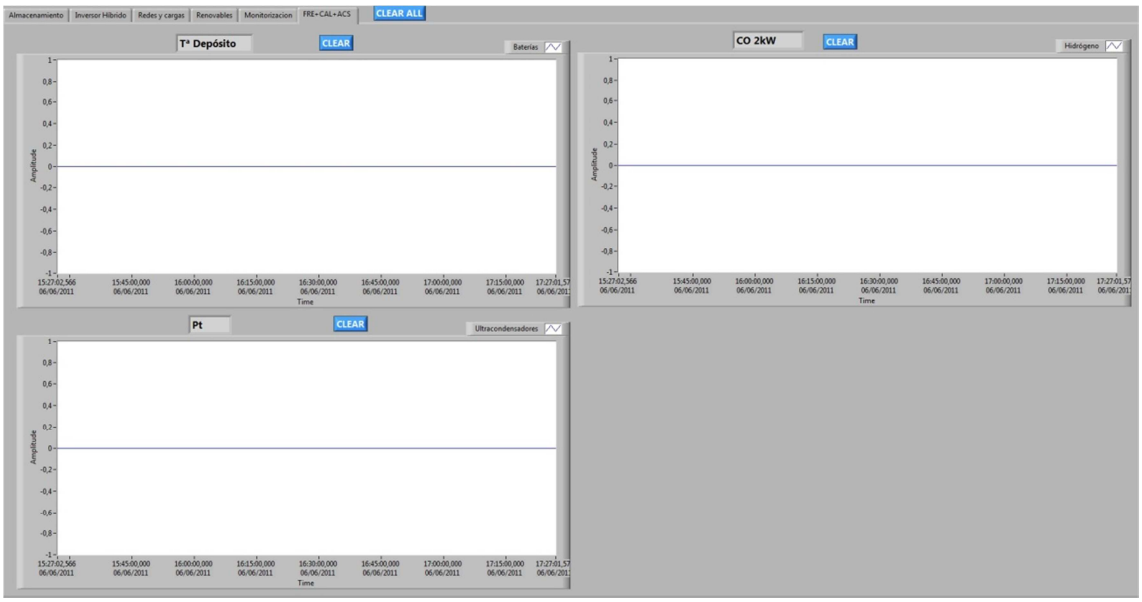

Figura 3.43: FRE+CAL+ACS

El usuario interactúa con esta serie de gráficas para elegir, en cada dispositivo, la variable a representar. Los datos a representar son de un total de 2h, pero puede borrarlos antes pulsando en el botón CLEAR asociado a cada gráfica, o borrarlos todos pulsando en el botón CLEAR ALL. También, al cambiar de una variable a otra dentro de un módulo, la gráfica se borrará automáticamente, para que no haya problemas de visualización debido a posible gran diferencia del rango de valores entre distintos datos.

Las variables a representar se pueden cambiar en cualquier momento, y, una vez elegida, el sistema comenzará automáticamente a dibujar los datos de esa variable producidos a partir de ese mismo momento.

Aunque una pestaña no esté activa, las gráficas en esa pestaña seguirán actualizándose, de modo que en todo momento se estarán actualizando 21 gráficas.

La idea es elegir las variables interesantes de visualizar e ir comprobando su evolución mirando en las distintas gráficas.

c) Diagrama de bloques

Como en todos los demás VIs, hay una parte de configuración (Figura 3.44) y otra de ejecución (Figura 3.45).

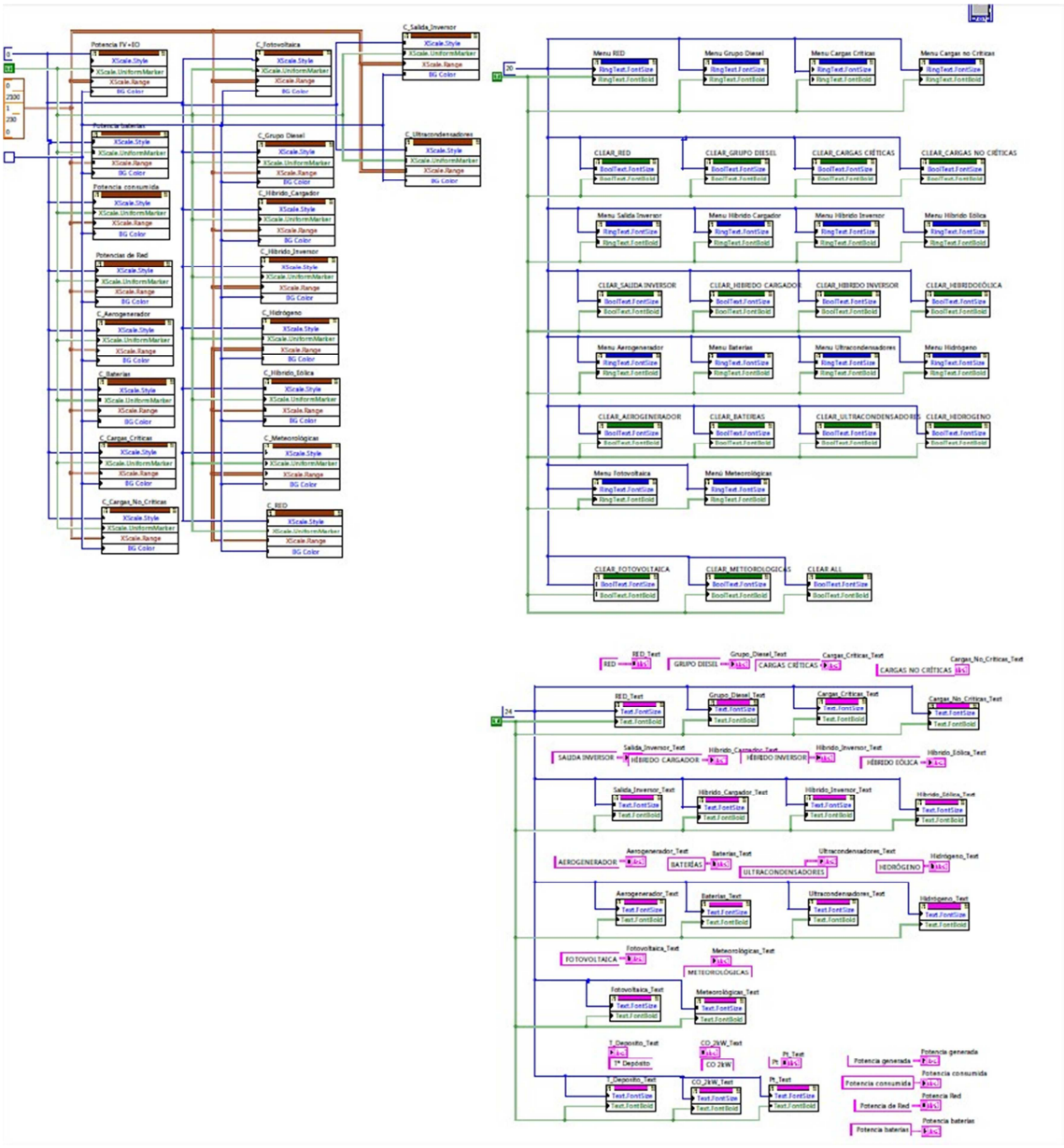

Figura 3.44: RealTimeCharts: configuración

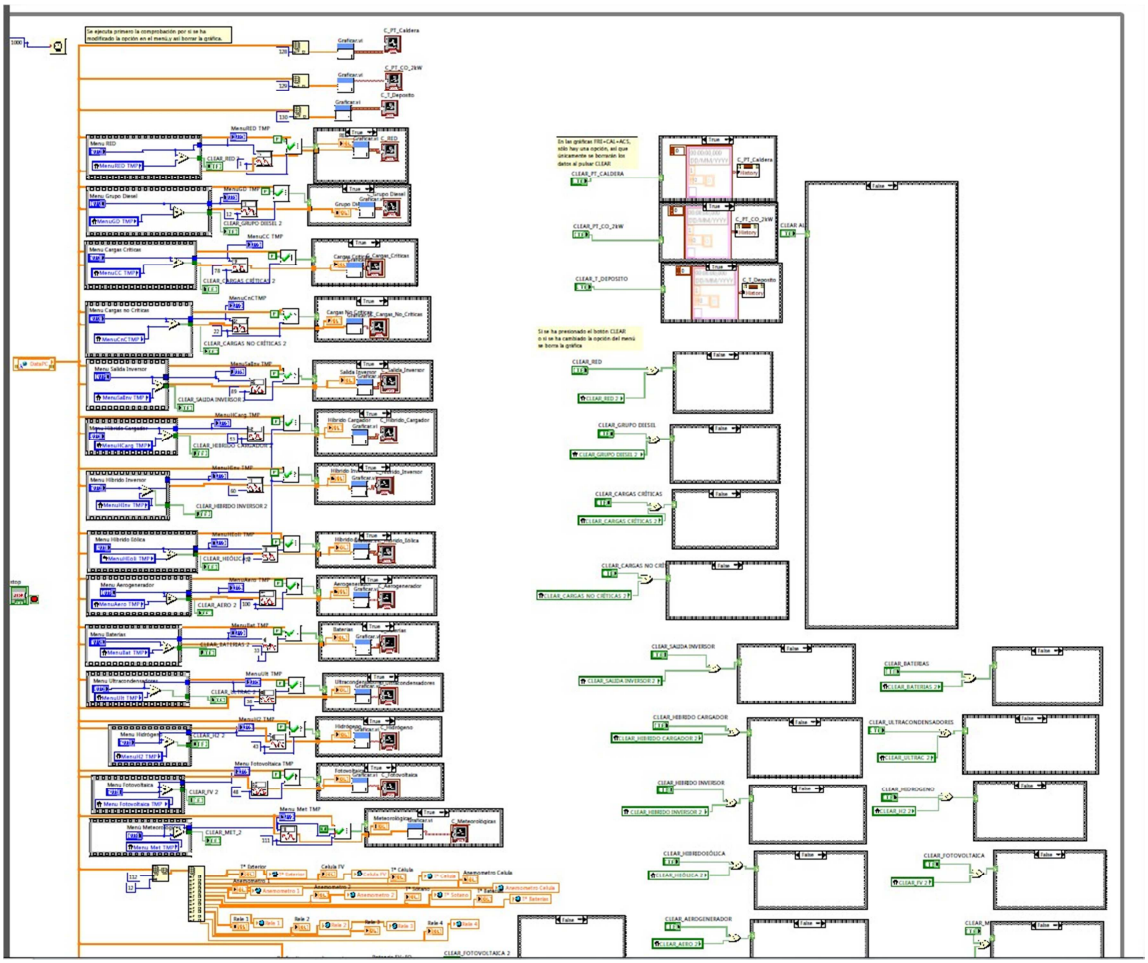

Figura 3.45: RealTimeCharts: ejecución

Una sección de configuración que merece ser comentada, por no haberse tratado hasta este momento, es la configuración de las gráficas, como se establece en la Figura 3.46.

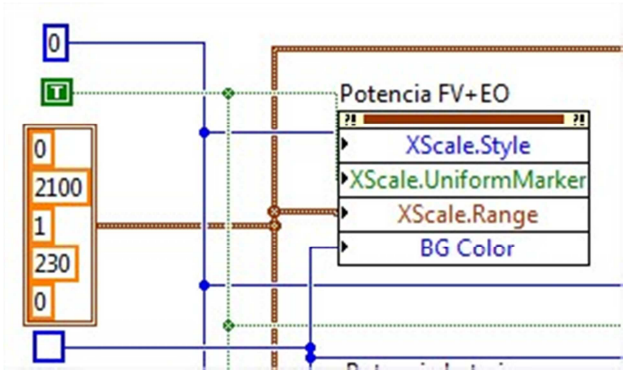

Figura 3.46: Configuración gráficas

En esta configuración, se especifica el estilo de la escala (0), la escala estará dividida uniformemente en valores (Uniform Marker), el rango de la escala (Range), y por último, una paleta de colores establecida en blanco para el color de fondo (BG Color).

El funcionamiento básico de las gráficas en tiempo real es el siguiente: a partir del array de datos recibidos y los valores en el menú, se establece el valor actual a representar. Además, hay que comprobar si se ha pulsado en botón CLEAR o se ha cambiado de variable a representar, para limpiar la gráfica antes de representar el nuevo valor. En caso de que los datos lleguen con error (valor del error en el dispositivo es distinto de 0), no se dibujará ningún valor en la gráfica, porque, en caso de que vengan con error, los datos no serán representativos, o quizás lleguen todo valores 0, y si se dibujasen, podrían confundirse con valores reales.

Los primero que se ejecuta dentro del bucle infinito (en el que cada iteración se da cada 1 segundo) es, para cada gráfica, la comprobación de si se ha cambiado de opción en el menú (Figura 3.47).

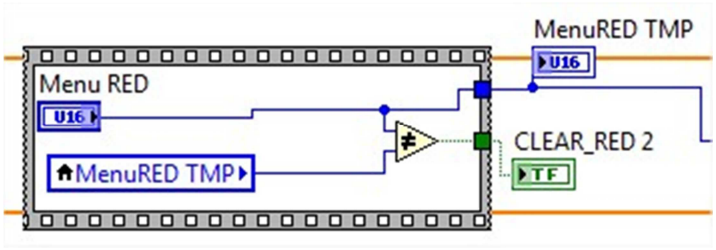

Figura 3.47: Comprobación de valor en el menú

Se comprueba si el valor del menú actual es el mismo que en la anterior iteración (dentro de la estructura, MenuRED TMP es una variable local que referencia a MenuRED TMP fuera de la estructura). Si es un valor distinto, la variable CLEAR_dispositivo 2 se establecerá a TRUE, indicando que hay que limpiar la gráfica.

Pero también hay que limpiar la gráfica si se ha pulsado en el botón CLEAR de la misma. Por lo tanto, se estable que si alguna de las dos variables vale TRUE (OR) se limpiará. Se puede ver en la Figura 3.48.

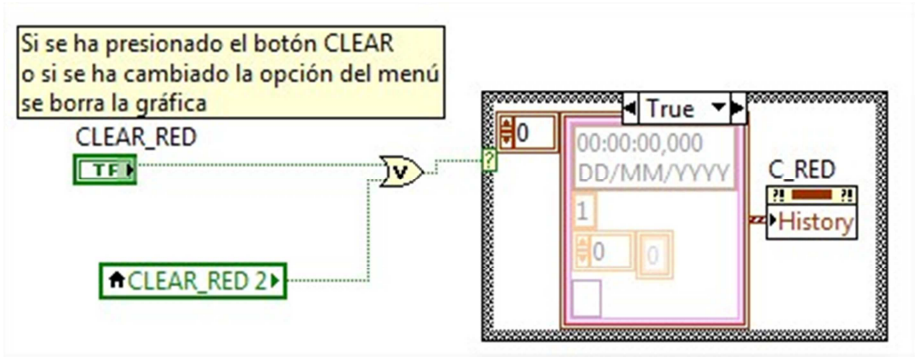

Figura 3.48: Limpieza de la gráfica RED

Hay que obtener el dato adecuado a la opción elegida en el menú. De eso se encarga la función creada MenuToChart. La función creada CheckError comprueba si se ha producido error de comunicación, para no dibujar los datos. También hay una función, creada por otra persona, que obtiene el valor de la gráfica a partir del valor doble, ya que LabVIEW requiere de una conversión, para poder establecer en qué momento se está

produciendo el dato, y dibujarlo con su respectiva hora y día. Se puede ver el funcionamiento de esta parte en la Figura 3.49.

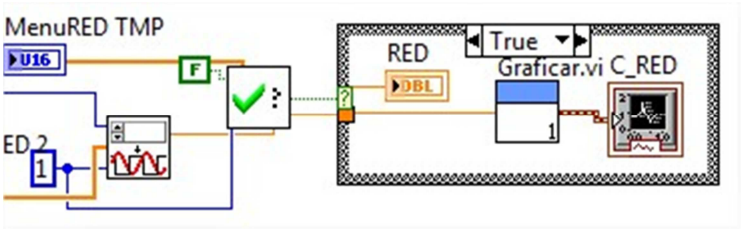

Figura 3.49: Graficar dato

Además, como se ha comentado previamente, este VI es el que asigna los valores a las variables globales correspondientes a las variables meteorológicas. Simplemente, extrae cada valor y lo escribe en la variable global (Figura 3.50).

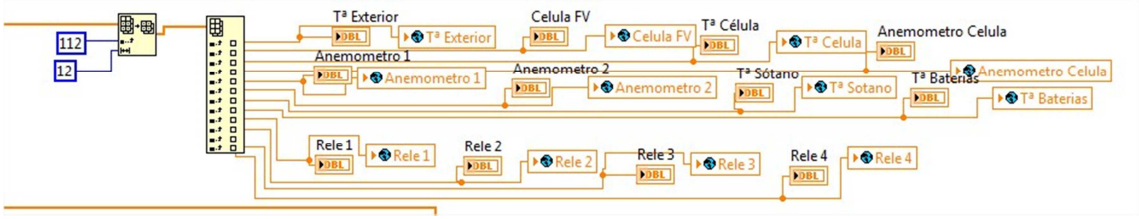

Figura 3.50: Asignación de valores en variables globales

El resto del VI no ha sido realizado por mí y no me corresponde explicarlo. De todos modos, su función es definir los datos adecuados a para las gráficas correspondientes a la pestaña de Monitorización.

d) Jerarquía del VI

Los SubVIs se pueden ver en la Figura 3.51. Para consultar su funcionamiento interno, dirigirse a la sección 3.2.10.

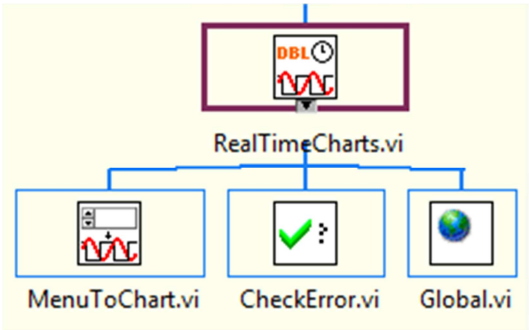

Figura 3.51: SubVIs utilizados

SubVIs utilizados:

- MenuToChart
- CheckError
- Global

3.2.6. Históricos

a) Descripción

Los históricos (HistoricosMicrorred.vi) proporcionan un menú a partir del cual se puede elegir qué datos de los almacenados se desean representar en una gráfica. Las gráficas en tiempo real nos daban un servicio similar a este, pero con una limitación de 2horas, sin embargo, el VI de Históricos está preparado para recoger datos con una variedad más amplia de selección.

Inicialmente no se planteó la elaboración de este programa, pero conforme avanzaba el proyecto quedó claro que debía existir alguna manera de acceder a los datos almacenados que fuesen más antiguos a 2horas.

Se pueden representar conjuntamente variables de distintos o de un mismo módulo, restringiendo los valores a una fecha establecida. El fin de este VI es representar de una manera amigable los datos almacenados en la base de datos, sin tener ninguna restricción de tiempo y pudiendo combinar en una misma gráfica los valores de distintas variables.

b) Panel Frontal

Icono: 

Representa la extracción de datos de la base de datos y su representación en una gráfica.

Se produjo una versión inicial de los Históricos que constaba de un menú en el que podíamos elegir las variables a representar y dos pestañas: una con una gráfica grande (Figura 3.52) y otra con cuatro gráficas más pequeñas (Figura 3.53)

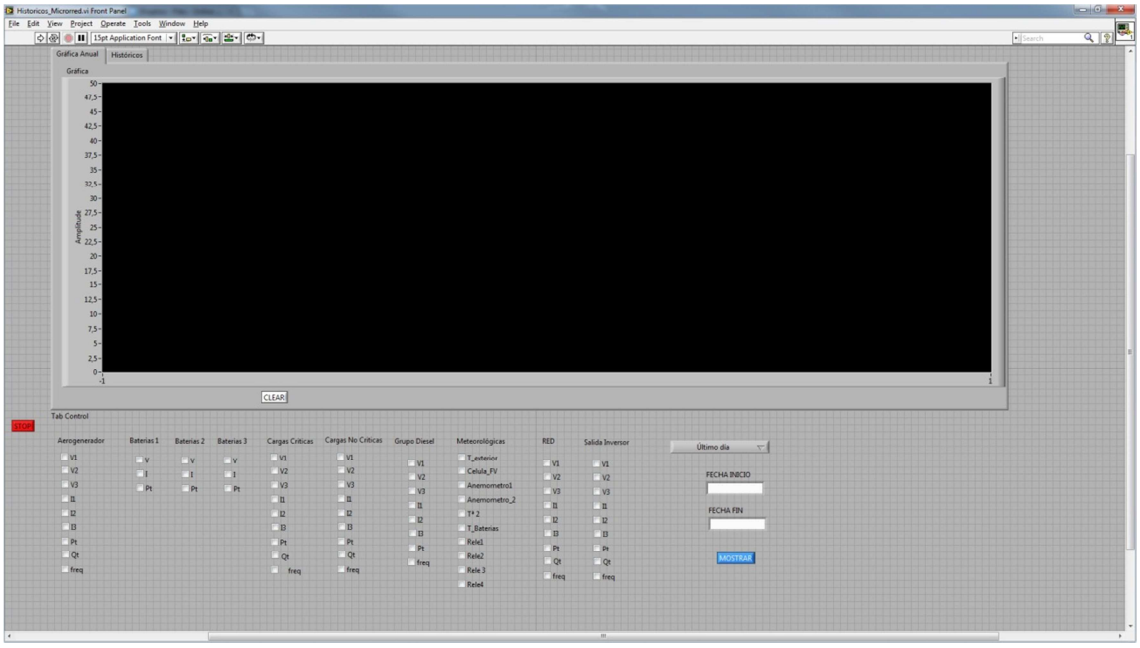

Figura 3.52: Gráfica grande y menú históricos

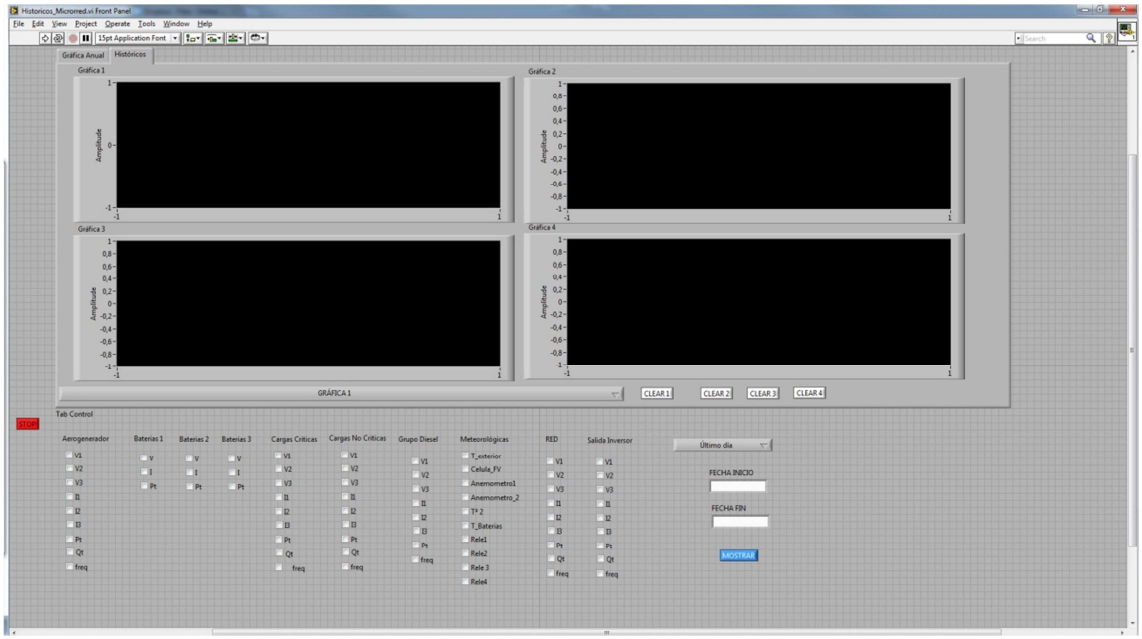

Figura 3.53: Gráficas pequeñas y menú históricos

Sin embargo, al igual que en todos los VIs anteriores, se consideró que no se estaba aprovechando al máximo las características de la pantalla, y que se podría modificar para ofrecer al usuario las gráficas más grandes posibles, ya que a mayor tamaño, mayor cantidad de datos podrían ser representados de una manera cómoda y sencilla de visualizar para el usuario.

Por tanto, se decidió que se añadiría una tercera pestaña con las opciones de selección de los datos. Así pues, las tres pestañas serían las siguientes:

- Gráfica anual (Figura 3.54)

  Gracias a su tamaño, esta gráfica está preparada para dibujar simultáneamente más de una variable de manera que se pueda apreciar claramente. Además, dispone de una leyenda en la que se actualizan al momento las variables representadas, con su color correspondiente en la gráfica. También está diseñada para albergar gran cantidad de datos, por ejemplo, de un año, por ello el nombre de la pestaña. Sin embargo, si en un momento dado sólo se desea representar una variable, aunque sea en un espacio de tiempo corto, también es la gráfica más adecuada debido a sus características. Por todo eso, se presupone que será la gráfica más utilizada, y se ha colocado en la primera pestaña, siendo lo primero que ve el usuario al acceder al VI. En cualquier momento podemos limpiar la gráfica clicando en el botón CLEAR. De todos modos, al representar unos datos automáticamente se limpiarán los anteriores.

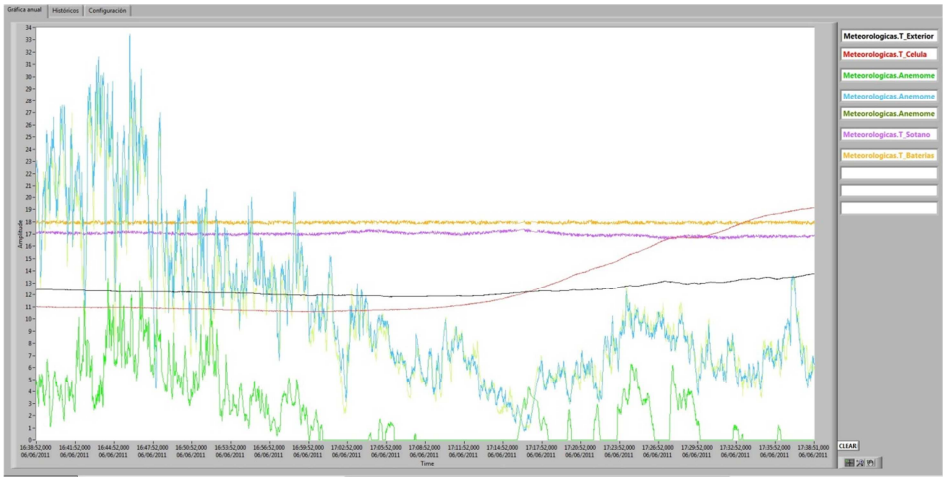

Figura 3.54: Gráfica anual

- Históricos (Figura 3.55)

En esta pestaña podemos apreciar cuatro gráficas de idéntico tamaño. Estas gráficas son apropiadas cuando deseas representar simultáneamente distintas variables en un mismo periodo de tiempo, pero la diferencia de sus valores es tal que en una sola gráfica, incluso siendo la anual, se apreciaría en un tamaño demasiado reducido cada una de las líneas. Por ejemplo, si un valor oscilase entre 0-1 y un segundo valor entre 200-250, representarlos en la misma gráfica haría confusa su lectura, pero si lo representamos en gráficas similares, ambas gráficas se moverán en el mismo espacio de tiempo por lo que podremos contrastar con mucha más facilidad estos valores. Sin embargo, estas gráficas no tienen leyenda, por lo que representar más de un valor no está recomendado, a no ser que sepamos diferenciarlos a simple vista. En cualquier momento podemos limpiar las gráficas clicando en el botón CLEAR correspondiente a cada gráfica. De todos modos, al representar unos datos automáticamente se limpiarán los anteriores.

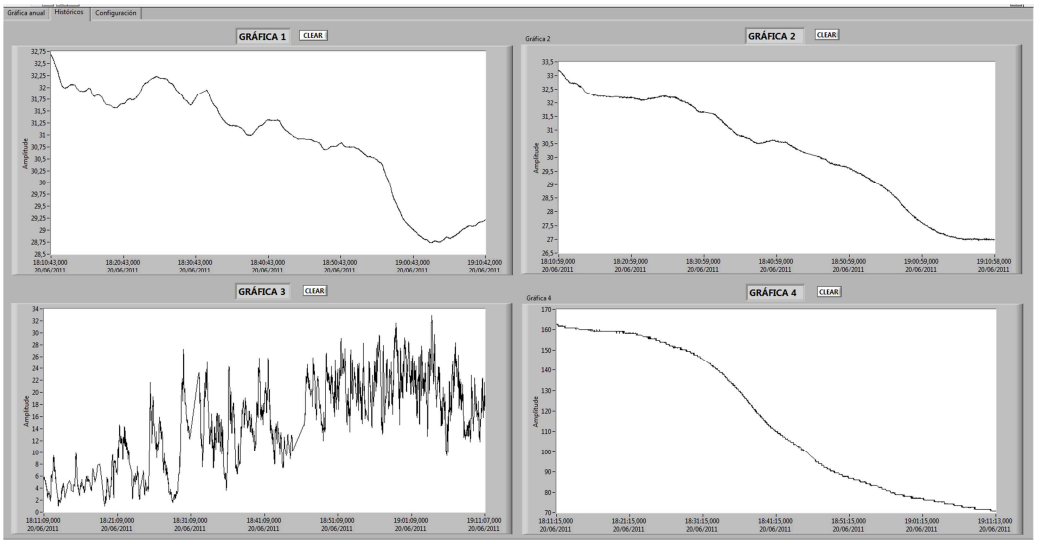

Figura 3.55: Históricos

- Configuración (Figura 3.56)

En esta pestaña se selecciona, por un lado, los valores a representar, y por otro, el espacio de tiempo en el que estamos interesados. Las posibles variables a seleccionar se encuentran agrupadas por dispositivo, y podemos seleccionar una o varias, de un mismo o distinto dispositivo. Hay que tener en cuenta que el límite de la leyenda en la gráfica anual es de 10 valores, por lo que, si seleccionamos más, no podremos ver el color asociado a esas variables, pudiendo inducir a error. De todos modos, se ha establecido un límite de 10 valores porque se ha considerado que el trazo de un número mayor llevaría a confusión.

En la parte inferior tenemos un menú en el que podemos elegir sobre qué gráfica representaremos los datos. Debido a que se prevé que la gráfica anual será la más utilizada, se ha establecido como gráfica por defecto. Así pues, solo tendremos que cambiar este menú si deseamos mostrar los datos en una de las gráficas pequeñas.

En la parte derecha tenemos los selectores del espacio de tiempo a representar. Hay un menú con las opciones básicas de forma que sea sencillo acceder a ellas. Dichos valores son: última hora, último día, última semana, último mes, último año y entre dos fechas. En todo momento se cuenta a partir del momento actual, así que, por ejemplo, si ahora son las 01:23:45h, seleccionar la última hora elegiría los datos desde las 00:23:45h hasta las 01:23:45h. Si seleccionamos la última opción, entre dos fechas, podremos elegir los momentos concretos desde los selectores de Fecha Inicio y Fecha Fin. Tras elegir el rango de tiempo deseado, los datos se graficarán pulsando en MOSTRAR. Hay que tener en cuenta que, aunque se haya conseguido un programa eficiente, dado que los datos se producen cada segundo, al elegir un rango elevado, tardará un poco en representarlo, teniendo en cuenta además que hace una serie de operaciones de medias con ellos, que serán explicadas en el Diagrama de Bloques.

En esta pestaña tenemos también dos botones: “CLEAR GRÁFICAS”, que limpiará todas las gráficas, y “CLEAR DATOS”, que limpiará los check box seleccionados.

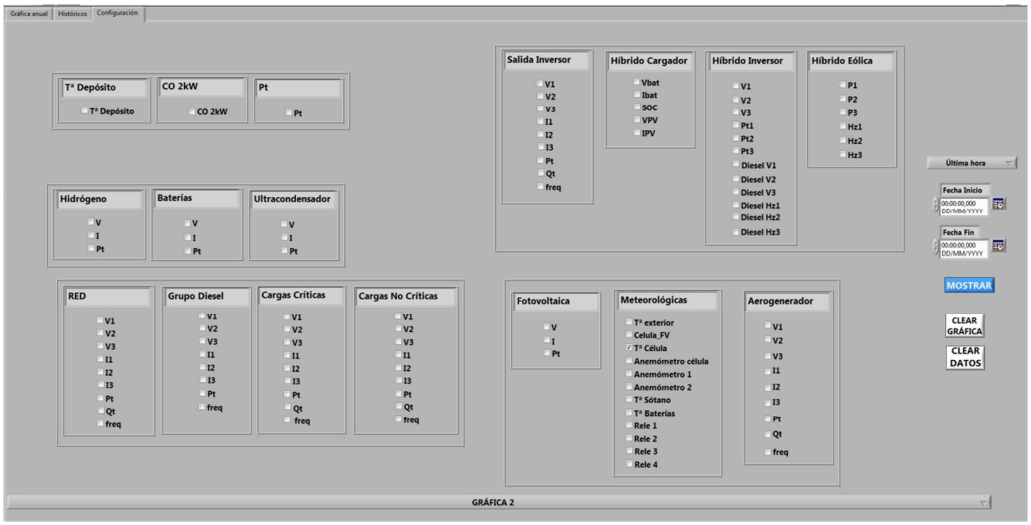

Figura 3.56: Históricos: configuración

c) Diagrama de bloques

Como en todos los VIs anteriores, hay una parte de configuración (Figura 3.57) y una de ejecución, que está dentro de un bucle infinito (Figura 3.58).

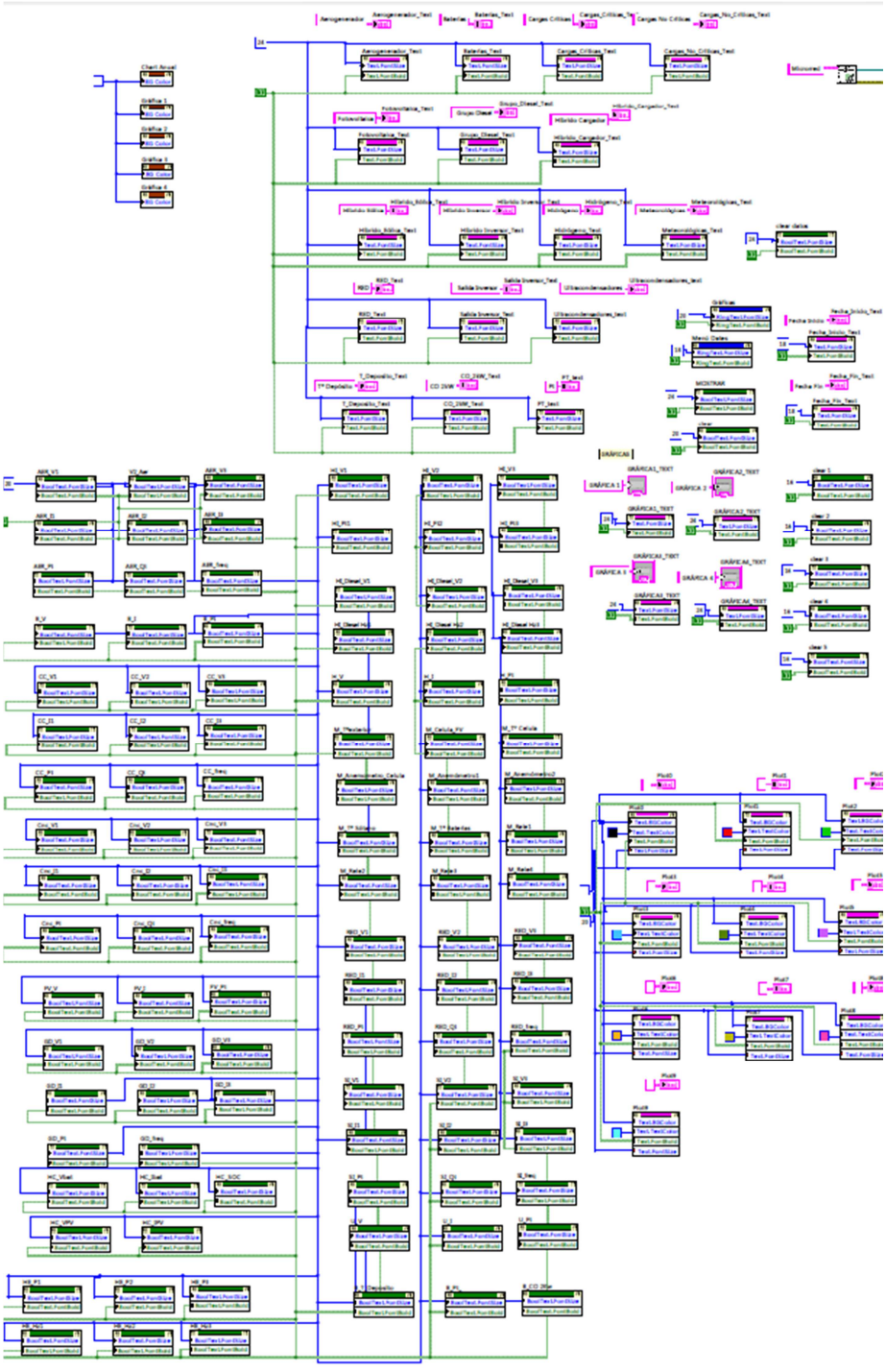

Figura 3.57: Históricos: configuración

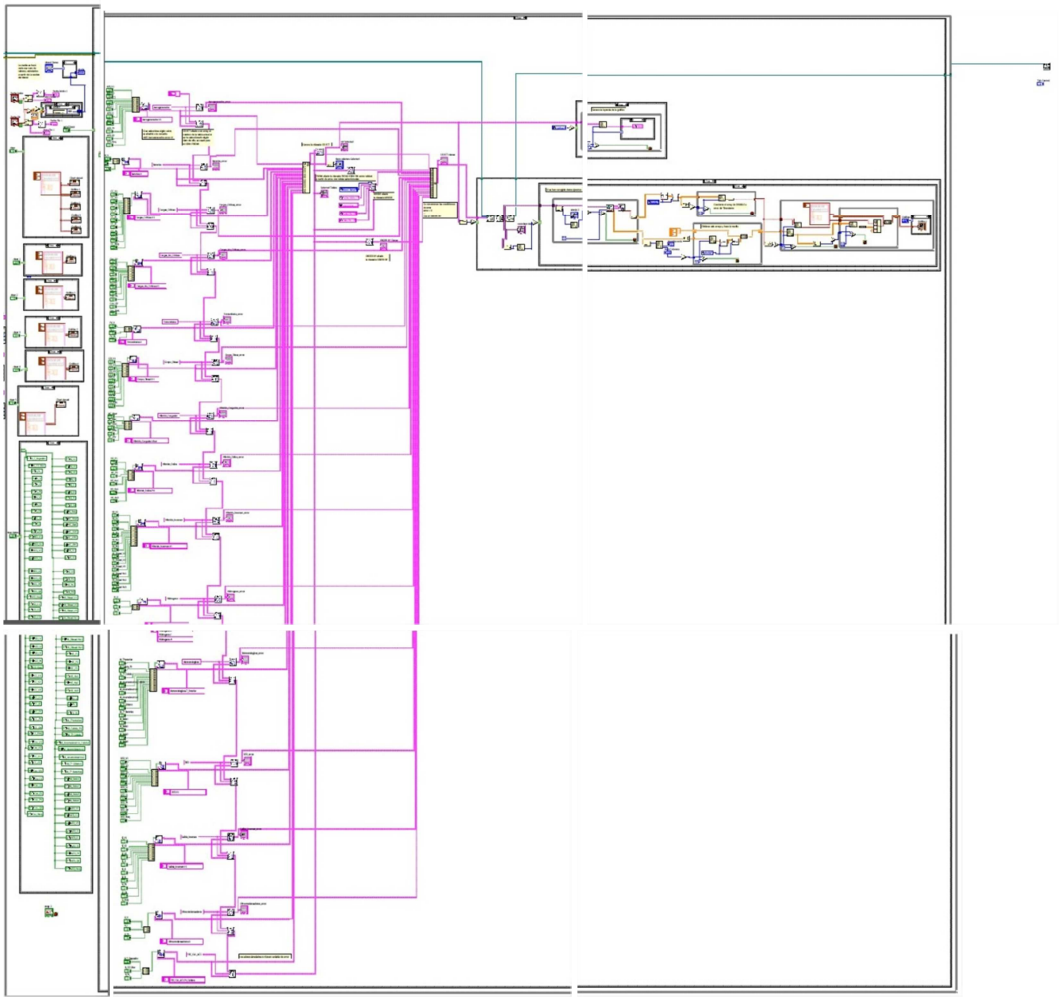

Figura 3.58: Históricos: ejecución

Respecto a la parte de configuración, cabe decir que se han configurado por adelantado los colores y tamaños de los letreros en la leyenda de la gráfica anual. Después, en la parte de iteración, se rellenarán según se vea conveniente. Estos colores se han establecido tratando de que fuesen muy diferentes entre sí, y coincidiendo con la configuración hecha para la propia gráfica anual, haciendo clic derecho sobre ella, Propiedades/Plots. En la Figura 3.59 podemos ver la configuración para uno de los valores de la leyenda. Para cada uno de los valores se ha establecido un color distinto en TextColor, dado por una paleta de colores.

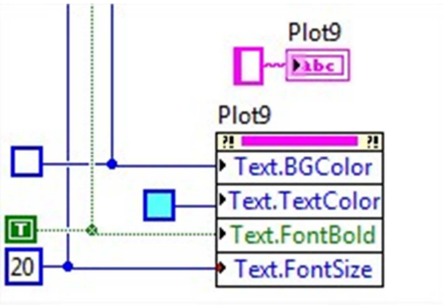

Figura 3.59: Configuración letrero leyenda

El resto de las configuraciones son similares a lo expuesto anteriormente.

Respecto al bucle de ejecución, antes de entrar en él, como se explicó en el Esquema de Comunicación (3.2.4), se inicia la conexión con la base de datos, y es a través de esta conexión por la que obtenemos los datos a representar. Dentro del bucle, podemos diferenciar una gran estructura case del resto. Lo programado en esta estructura solo se ejecutará cuando pulsemos el botón "MOSTRAR". Fuera de esta estructura, tenemos por un lado los botones que nos limpiarán las gráficas y los checkbox, y por otro una serie de variables que almacenarán el rango de tiempo deseado, y el valor a partir del cual ser hará la media. Podemos verlo en la Figura 3.60.

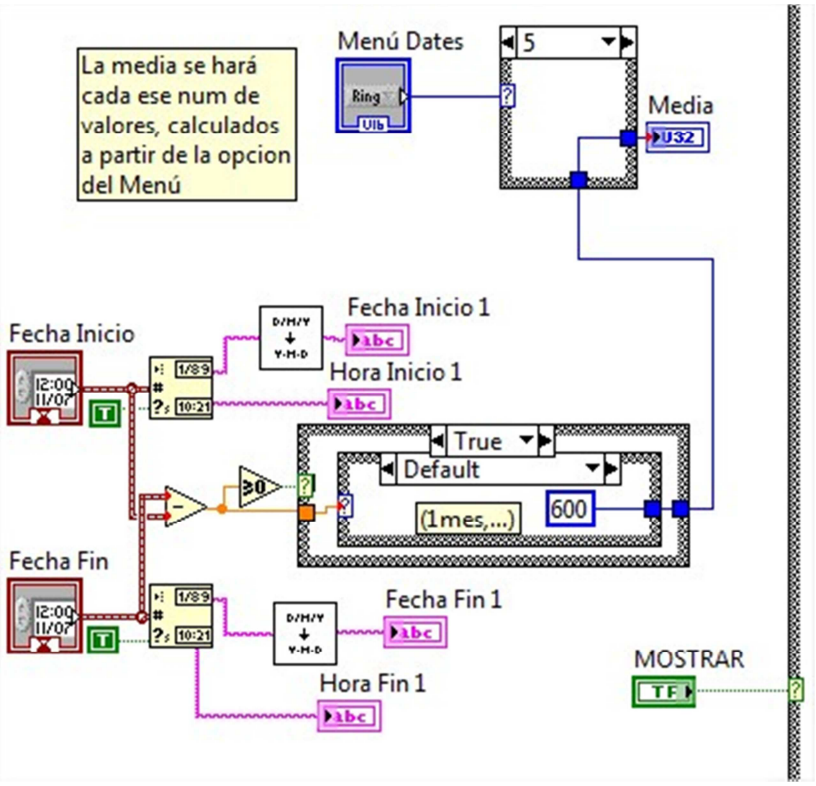

Figura 3.60: Históricos, selección de rango de tiempo y media

Al hacer la selección de los datos, pueden llegar a ser tantos que sea imposible, ineficiente e inviable el representarlos todos en la gráfica. Por ello, se pensó que dependiendo del rango de tiempo a representar, se haría la media de los datos según una cantidad de valores.

Tiempo	Frecuencia	Media	Valores
Hora	1s	1	3600
Día	5s	5	17280
Semana	15s	15	40320
Mes	1min	60	43200
Año	10min	600	52560

Así, si se eligen los datos de la última hora, la frecuencia de datos será de 1 segundo, lo que significará hacer la media por cada valor, o lo que es lo mismo, no realizar media y representarlos todos. Por ello, habrá un total de 3600 valores.

En cambio, si se selecciona del último día, por ejemplo, se agruparán los datos de cada 5 segundos y se hará la media entre ellos. En total se representarán 17280 datos.

Para los cálculos de la semana, agrupándolos por cada 15 segundos, se estima un número de datos de 40320.

En el caso de querer representar los datos de un mes, se agruparán por minutos, lo que nos dará un total de 43200 datos en la gráfica.

Por último, en caso de representar un año, los datos serán agrupados por cada 10 minutos, por lo que la media se hará por cada 600 datos (número de datos en 1 minuto) y habrá un total de 52560 valores a representar.

Esta tabla nos sirve calcular en valor de la variable media si seleccionamos una de las opciones del menú que no sea la última. Sin embargo, al elegir “Entre dos fechas”, surge el problema de que podríamos establecer fechas muy separadas o muy juntas. Para resolverlo, se calcula la diferencia entre “Fecha Fin” y “Fecha Inicio”. Si elegimos una fecha de finalización anterior a la fecha de inicio, nos será mostrado un cartel de error con el mensaje “No hay datos disponibles”. En el caso de que las fechas sean adecuadas, a partir de la diferencia entre ellas, se establece la media. Si entre ellas hay hasta un total de 3600 segundos de diferencia, el sistema se comportará haciendo medias con la misma frecuencia de datos que con la opción “Última hora”. Así sucede por cada rango de valores, teniendo el rango por defecto el valor para la media 600, pues se considera que abarca todas las cantidades que son mayores a las recogidas por las demás opciones.

Respecto a lo programado dentro de la estructura, es decir, las funciones que realmente recogen y representan los datos, ha cambiado profundamente desde los inicios del proyecto. En un principio, se recogían los datos de todos los módulos según los tiempos especificados y posteriormente sólo se representaban los seleccionados. Sin embargo, queda claro que esta técnica, aunque fuese eficiente por el poco tiempo que empleaba en realizar la petición a la base de datos, era totalmente ineficaz una vez que el volumen de datos crecía.

Después se realizó una segunda versión, pensada para que la selección de los datos fuese muy rápida, de modo que los datos de los distintos dispositivos eran recogidos por separado, sólo en caso de que se hubiese seleccionado alguna variable de dicho dispositivo. Aunque la selección de los datos era adecuada, LabVIEW presentaba muchos problemas a la hora de juntar los distintos datos en una sola gráfica, de modo que fuesen coherentes.

Por último, se ideó la solución final, que mejoraba a las anteriores no solo en tiempo de ejecución, si no también en cantidad de espacio ocupado, por lo que resultaba mucho más eficiente. En esta versión se aprovechó más la capacidad de la base de datos

que la de LabVIEW. SQL Server es capaz de responder a consultar muy complejas, y dado que era capaz de idear una consulta compleja pero efectiva, me decanté por esa opción. Sin embargo, esto resulta en que la comprensión del código para alguien que no tenga un cierto dominio en bases de datos sea menor.

Básicamente, la mayoría del código sirve para generar una consulta para SQL Server muy extensa, de modo que devuelve los datos adecuados que son tratados en una serie de bucles y por último, representados.

- Generación de la consulta SQL

La consulta SQL generada devolverá los datos adecuados de la base de datos, utilizando la conexión ya establecida. La consulta final sigue un esquema como el siguiente:

```
SELECT Fecha, Hora, variables_seleccionadas
FROM dispositivos_seleccionados
WHERE fechas_adecuadas AND no_hay_error
ORDER BY Fecha, Hora
```

Es decir, selecciona la fecha y la hora, además de todas las variables seleccionadas en los check_box. Estos datos los recogerá de las relaciones en la base de datos correspondientes a los dispositivos de los cuales se haya seleccionado al menos un campo. La selección de los datos estará restringida a que se hayan producido entre unas fechas determinadas. Además, no se seleccionarán datos cuya variable de error marque error, si no que serán obviados, para evitar falsos datos o datos erróneos en la representación. Por último, se ordenarán esos datos por la fecha y la hora, de modo que los primeros datos a representar sean los más antiguos.

Para la generación de esta consulta se han creado varias funciones propias, almacenadas en la librería creada SQLToolsLLB, cuyo funcionamiento interno se puede ver en la sección 3.2.10.

Lo primero, es saber qué datos ha seleccionado el usuario, y a qué dispositivo pertenecen, de forma que se puedan seleccionar esos valores y únicamente esos. Cada checkbox que esté seleccionado tendrá como valor TRUE. A función creada SelectedArray.vi devuelve, a partir de los valores de los check box, y de la lista de variables, un array con las columnas seleccionadas. Podemos ver el ejemplo con las Baterías en la Figura 3.61.

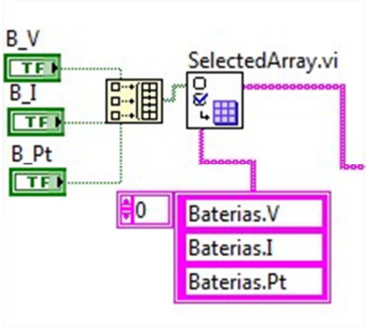

Figura 3.61: Columnas seleccionadas del dispositivo Baterías

Ahora tenemos un array por cada dispositivo con las variables seleccionadas. Ese array, llamado SelectedColumns, se usa en distintas funciones. Primero, se pasa como argumento a la función creada WHEREerrorClause.vi, que añadirá a la consulta la parte del WHERE en la que se especifica que no se deben seleccionar datos erróneos de ese dispositivo. Hay que tener en cuenta que los datos simulados no tienen variable de error, así que no tendrán en cuenta esta función.

La misma información se pasará también a la función creada SelectedTables.vi, de modo que al finalizar, nos devuelva un array con la lista de dispositivos distintos de los que se desea extraer datos.

A partir de las columnas seleccionadas, la función creada SELECTclause.vi generará la cláusula SELECT. Con ese mismo dato, la función creada FROMclause.vi generará la cláusula FROM de la consulta. La función WHEREclause.vi también toma como entrada ese dato, además de todas las variables correspondientes a la elección del rango de tiempo: “Menú date”, “Fecha Inicio”, “Hora Inicio”, “Fecha Fin”, “Hora Fin”. Con todo ello genera la cláusula WHERE. Por último, también haciendo uso del dato que indica los dispositivos seleccionados, ORDERBYclause.vi generará la cláusula ORDERBY. Todas las cláusulas generadas serán concatenadas de forma que, al final, se construya la consulta SQL adecuadamente.

Antes de usar esa consulta para seleccionar los datos, se ejecutará un bucle que rellenará adecuadamente la leyenda de la gráfica anual a partir de las variables seleccionadas. (Figura 3.62)

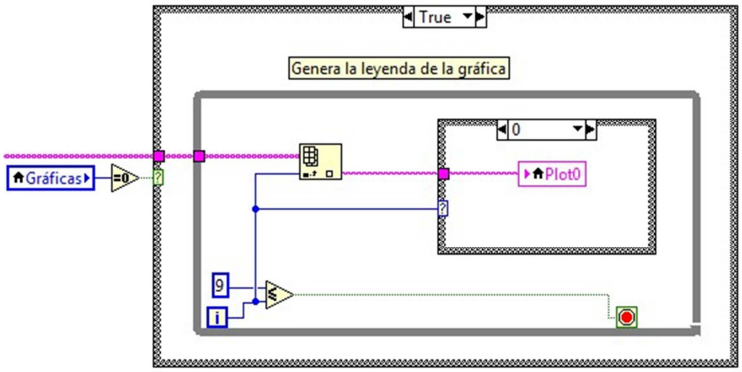

Figura 3.62: Generación de la leyenda

- Selección, tratamiento y representación de los datos

Una vez que ya se tiene la consulta SQL, hay que seleccionar los datos de la base de datos. En caso de que no se haya seleccionado ninguna variable, un mensaje de error avisará de “Ningún elemento seleccionado”. Si se ha comprobado que se han seleccionado datos, se extraerán de la base de datos mediante las funciones propias de LabVIEW: DBToolsExecuteQuery, que ejecutará la consulta creada, DBToolsFetchRecordsetData, que extraerá los datos, y DBToolsFreeObject que destruye la referencia a los datos, de modo que la conexión queda libre. A continuación se comprueba si los datos devueltos tienen un tamaño mayor que 0, es decir, si la consulta ha devuelto algún dato. Este proceso se ve en la Figura 3.63.

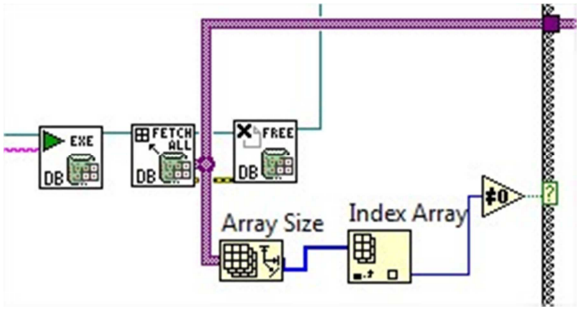

Figura 3.63: Extracción de datos de la base de datos

En caso de que el tamaño sea 0, se producirá un mensaje de error con “No hay datos disponibles” y terminará la ejecución. En caso contrario, se comenzará con el tratamiento de los datos. El primer bucle a ejecutar se ve en la Figura 3.64.

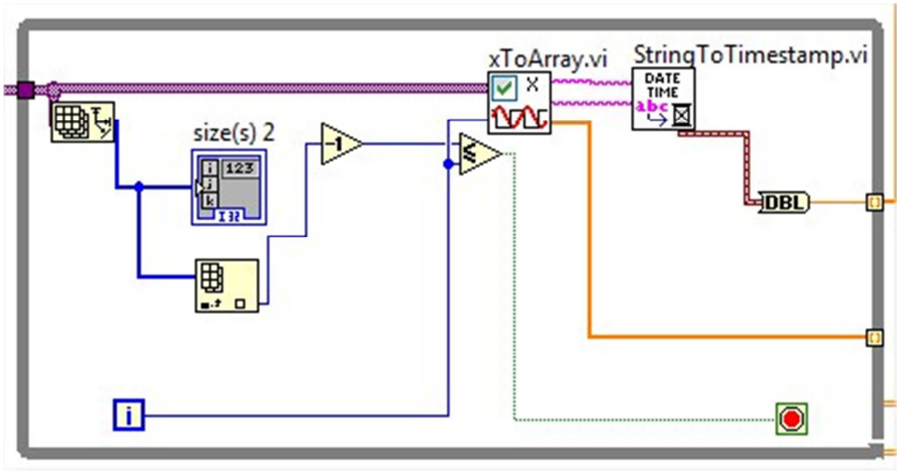

Figura 3.64: Tratamiento inicial de los datos

En este bucle, se extraen los datos devueltos por la base de datos, de forma que, haciendo uso de las funciones creadas xToArray.vi y StringToTimestamp.vi, se generan dos estructuras de datos: un array de dobles con las fechas y horas, y una matriz con todos los datos de todas las variables seleccionadas. Después de eso, cada estructura de datos es tratada por un bucle diferente.

El array de dobles con las fechas y horas (timestamps) sigue el tratamiento de la Figura 3.65.

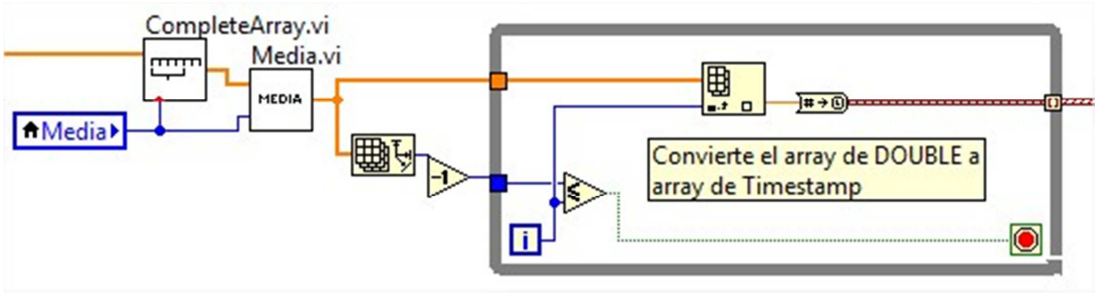

Figura 3.65: Tratamiento de timestamps

Se hace uso de la variable Media asignada al principio, y dos funciones creadas. CompleteArray.vi completa un array de entrada para que su número de elementos cumpla Size(CompletedArray) % Media =0 Es decir, el resto de dividir el tamaño entre la media sea 0. De esta forma, si se quiere hacer media cada n valores, el número de valores en el array será siempre un múltiplo exacto de n, de manera que no se generen falsos datos. A continuación, el array completo se pasa como argumento a Media.vi, que devuelve un array con los datos en los que ya se ha hecho la media. Por último, se ejecuta un bucle en el que se convierte cada posición de doble a timestamp.

La matriz con los datos sigue un camino diferente, como se puede ver en la Figura 3.66.

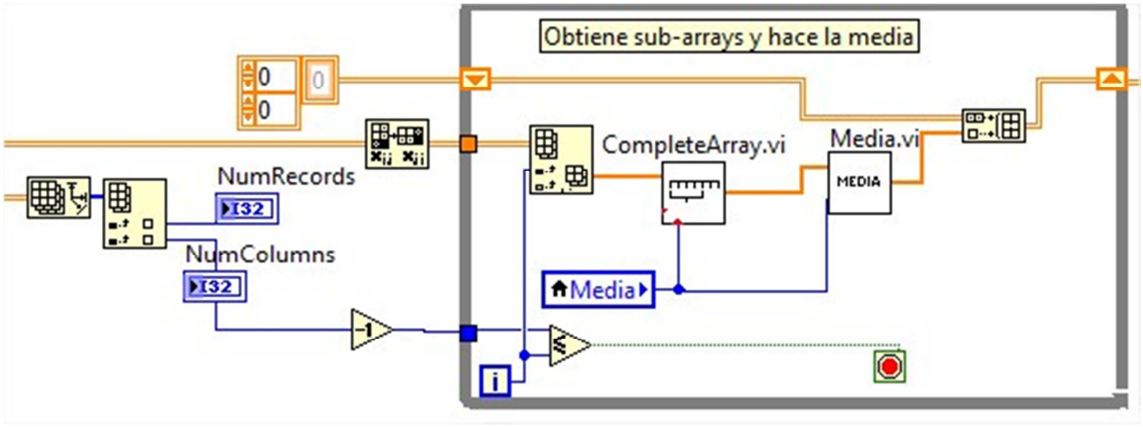

Figura 3.66: Tratamiento de matriz de valores

Primero, se transpone la matriz, de modo que cada fila corresponda a los valores de una variable. Así, dentro del bucle, para cada fila, completaremos su array y haremos la media. Este bucle devuelve una matriz en la que cada fila son los valores, una vez hechas las medias, de cada una de las variables.

Por último, tanto el array de timestamps con los valores medios, como la matriz de los valores medios de las variables entran en un bucle, como se ve en la Figura 3.67.

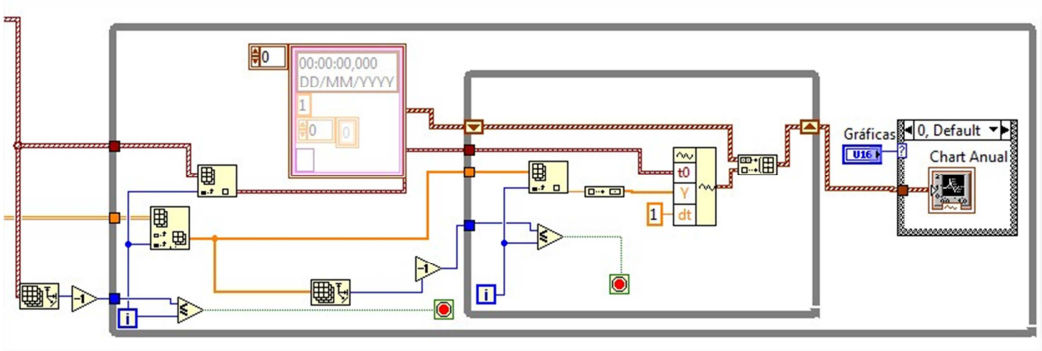

Figura 3.67: Representación de los datos

Este bucle se ejecuta para cada una de las posiciones del array de timestamps. En cada iteración, se coge una posición del array de timestamps, y la columna de esa misma posición, que tendrá los valores de todos los dispositivos para ese momento. En el bucle interior, para cada valor de la columna se genera un dato a representar en la gráfica. La gráfica en la que se representará vendrá dado según el valor del menú Gráficas. El tratamiento de datos se ve en la Figura 3.68.

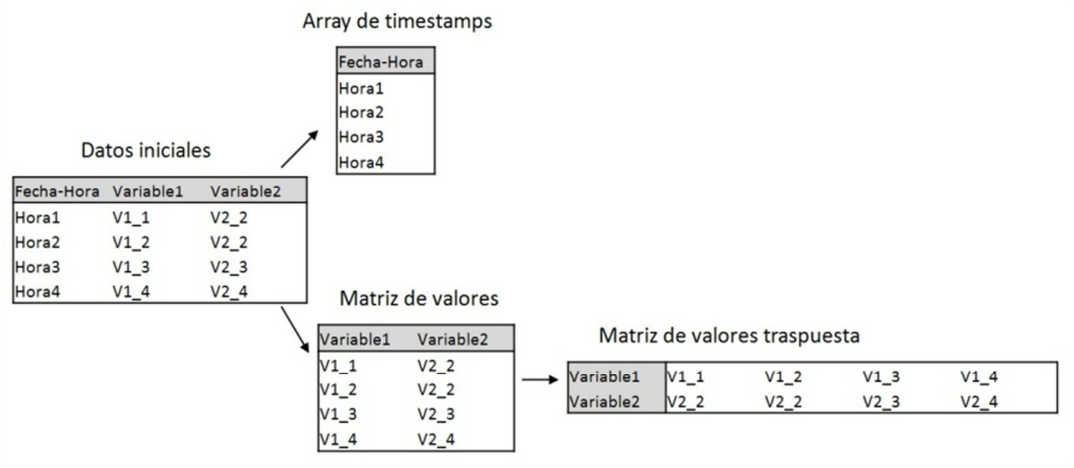

Figura 3.68: Esquema del tratamiento de datos

d) Jerarquía del VI

Los SubVIs se pueden ver en la Figura 3.69. Para consultar su funcionamiento interno, dirigirse a la sección 3.2.10.

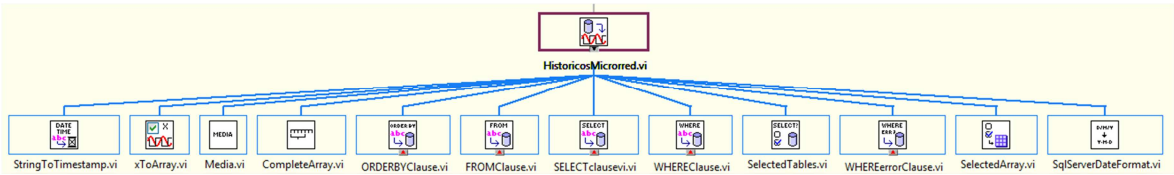

Figura 3.69: SubVIs utilizados

SubVIs utilizados:

- StringToTimestamp
- xToArray
- Media
- CompleteArray
- ORDERBYclause
- FROMclause
- SELECTclause
- WHEREclause
- SelectedTables
- WHEREerrorClause
- SelectedArray
- SqlServerDateFormat

3.2.7. Selección de datos

a) Descripción

El selector de datos (SelectData.vi) proporciona un menú a partir del cual se puede elegir qué datos de los almacenados nos resultan interesantes, bien para guardar el array de datos, bien para generar un fichero de texto .txt o un fichero de Microsoft Excel .xls. También interesa ejecutar algún determinado VI con unos determinados datos.

Inicialmente no se planteó la elaboración de este programa, si no que surgió como un complemento del VI de Históricos. En dicho VI se pueden ver los datos representados gráficamente, pero no podemos almacenarlos en un documento para estudiarlos detenidamente y de manera exacta. Además, se planteó la cuestión de ejecutar algún determinado VI que hiciese cálculos sobre unos datos elegidos determinados, y los VI ya creados no incluían esa opción.

El VI de selección de datos se basa en el de Históricos, por lo que su funcionamiento es similar. Únicamente se han añadido algunas opciones para las nuevas funcionalidades implementadas. En un principio, puede parecer lógico que la elaboración de informes de datos, ya sean en documento de texto o en formato Excel, surgiese también a partir de los Históricos.

Sin embargo, quedaba la cuestión de ejecutar algún subVI sobre los datos seleccionados, y esto último no era compatible con los históricos, ya que mientras se ejecutase ese subVI no se podían consultar las gráficas, por lo que se perdía funcionalidad.

Así pues, se decidió crear este VI de selección de datos con el fin de tratarlos o representarlos de algún modo. Al igual que en los históricos, se pueden seleccionar variables de distintos o de un mismo dispositivo.

b) Panel Frontal

Icono: 

Representa la extracción de datos de la base de datos y su escritura en un informe.

La interfaz con el usuario se puede apreciar en la Figura 3.70.

Figura 3.70: Selección de datos

El panel frontal está basado en el de Históricos, con la diferencia de que éste solo contiene una pestaña, la que correspondía a Configuración en Historicos.vi. Se ha añadido el menú que se puede ver en la Figura 3.71.

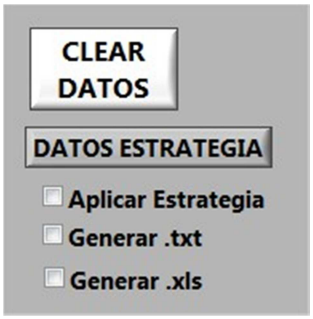

Figura 3.71: Menú de selección

Este menú nos proporciona las funcionalidades del VI. Por un lado, CLEAR DATOS deseleccionará todos los checkbox.

El botón DATOS ESTRATEGIA seleccionará por defecto los datos que son interesantes para ejecutar el subVI. Estos datos son los siguientes:

- Pt Cargas No Críticas
- Pt Fotovoltaica
- Vbat Híbrido Cargador
- Ibat Híbrido Cargador
- SOC Híbrido Cargador
- Pt1 Híbrido Eólica
- Pt2 Híbrido Eólica
- Pt3 Híbrido Eólica

- Pt RED
- Pt Cargas Críticas
- Tª Exterior Meteorológicas
- Célula FV Meteorológicas

Si seleccionamos el checkbox "Aplicar Estrategia", se ejecutará el subVI correspondiente con los datos seleccionados. El check box "Generar .txt" generará un documento de texto con los datos seleccionados. El check box "Generar .xls" generará un documento de Microsoft Excel con los datos seleccionados.

Elijamos la opción que elijamos, los datos aparecerán en tres estructuras de datos distintas, como se puede ver en la Figura 3.72.

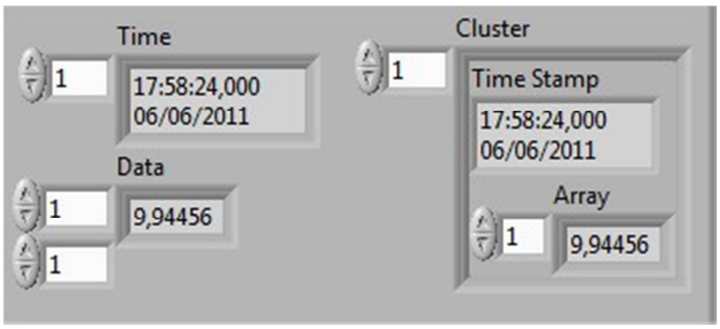

Figura 3.72: Estructuras de datos con los datos seleccionados

En la estructura "Time" podremos ver un array con los timestamp (Fecha y Hora) seleccionados. En la estructura "Data" podemos ver una matriz con los datos seleccionados.
En la estructura "Cluster" podremos ver un clúster que combina las dos estructuras anteriores.
Todas ellas representan los mismos datos, pero en diferente formato para elegir las que sean de nuestra necesidad.

Si seleccionamos generar algún tipo de documento, se rellenarán las casillas correspondientes con la ruta de dicho documento, como se puede ver en la Figura 3.73.

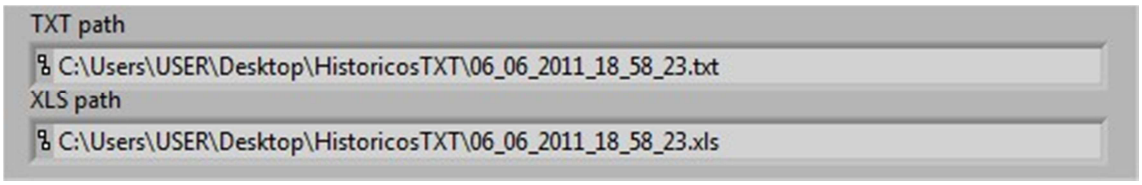

Figura 3.73: Ruta de los documentos

Por defecto se almacenan en una carpeta alojada en el escritorio y como nombre llevan la fecha y hora actual.

c) Diagrama de bloques

La selección y el tratamiento de los datos sucede igual que en Históricos.vi (3.2.6). Lo único que cambia son las nuevas funcionalidades, así que, una vez que tenemos los arrays de datos correspondientes, hay tres bucles para cada una de las funcionalidades.

Si deseamos aplicar la estrategia se ejecutará lo correspondiente a la Figura 3.74.

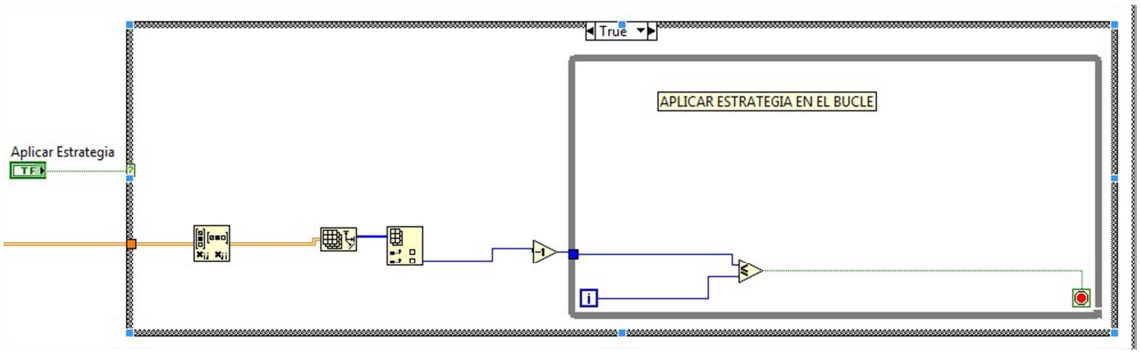

Figura 3.74: Ejecución de estrategia

Es decir, si hemos seleccionado el check box, se ejecutará este bucle, en el que se han traspuesto los datos de forma que cada fila corresponda a una variable. Dentro del bucle no se ejecuta nada, ya que no me corresponde a el planteamiento y la ejecución de esa estrategia, únicamente dar la base para que se pueda ejecutar sobre unos datos seleccionados.

Si deseamos generar un documento de texto .txt se ejecutará el código correspondiente a la Figura 3.75. En la primera línea aparecerán los nombres de las variables (en primer lugar Fecha y Hora), separados por tabuladores. El resto de datos también aparecen separados por tabuladores.

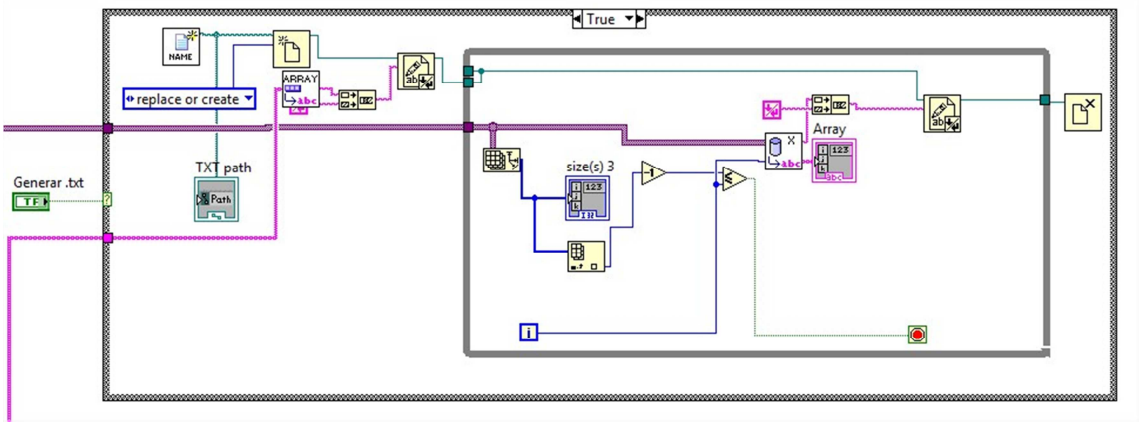

Figura 3.75: Generación de documento .txt

La función creada CreateFilename genera el nombre del documento y devuelve la ruta de dicho documento. La función creada ArrayToString genera una cadena de caracteres separada por tabuladores a partir del array con los nombres de los datos seleccionados. Después se ejecuta un bucle para cada grupo de datos de un momento

específico, y la función creada xToString generará una cadena de caracteres con los datos separados por tabuladores. Cada cadena de esas será escrita en un documento .txt.

Si deseamos generar un documento de Microsoft Excel se ejecutará el código correspondiente a la Figura 3.76.

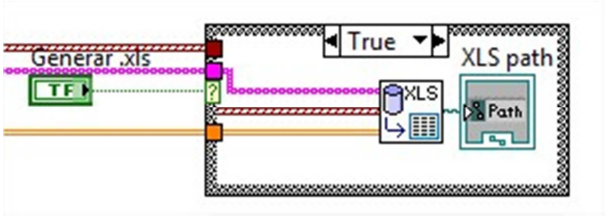

Figura 3.76: Generación de documento .xls

La función creada WriteToXLS genera un documento XLS y devuelve la ruta de dicho documento.

d) Jerarquía del VI

Los SubVIs se pueden ver en la Figura 3.77. Para consultar su funcionamiento interno, dirigirse a la sección 3.2.10.

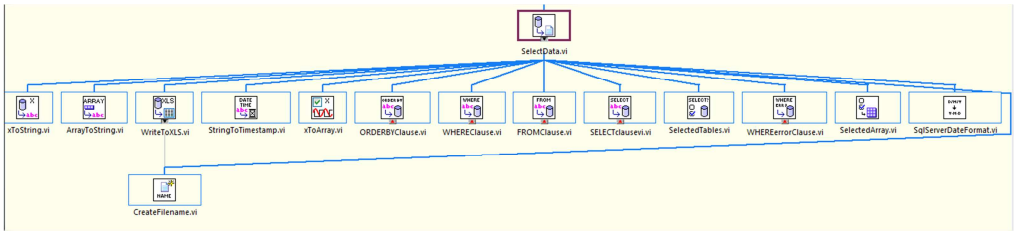

Figura 3.77: SubVIs utilizados

SubVIs utilizados:

- xToString
- ArrayToString
- WriteToXLS
  - CreateFilename
- StringToTimestamp
- xToArray
- ORDERBYclause
- WHEREclause
- FROMclause
- SELECTclause
- SelectedTables
- WHEREerrorClause
- SelectedArray
- SqlServerDateFormat
- CreateFilename

3.2.8. FTP

a) Descripción

El recolector de FTP (RecolectarFTP.vi) proporciona una seguridad extra frente a la posible pérdida de los datos. El PXi ha sido programado de manera que, los datos que recoja, además de enviarlos al PC, los almacene en un fichero de tipo TDMS. Estos ficheros que están almacenados en el PXi se pueden recoger desde el PC vía FTP. Es decir, en caso de que el PC no haya podido recibir datos (fallo de corriente, reinicio inesperado,...) puede acceder al fichero específico TDMS y a partir de esos datos rellenar la base de datos.

Se entiende que el PXi es un equipo que nunca se apaga, pero el PC no será así, y de ahí se generó la idea de este VI, cuyo último objetivo es la no pérdida de datos.

Se ejecuta automáticamente y no necesita intervención por parte del usuario para su correcto funcionamiento, excepto en una selección inicial de modo de funcionamiento. En un principio, una vez cada hora comprobará si faltan datos, y, en caso afirmativo, rescatará los ficheros TDMS correspondientes e insertará los datos en la base de datos.

b) Panel Frontal

Icono: 

Representa la recolección de ficheros mediante FTP.

El Panel Frontal del recolector FTP no contiene nada de interés de ser comentado. La razón es que no se trata de un VI de consulta, si no de uno que, en principio, se ejecutará automáticamente y sin intervención.

De todos modos, al ejecutar el VI aparecerá una ventana emergente con el menú que se puede ver en la Figura 3.78.

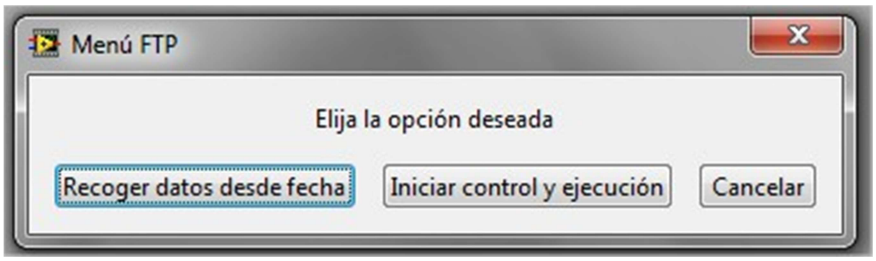

Figura 3.78: Menú inicial recolector FTP

Este menú presenta tres opciones:

- Recoger datos desde fecha

Con esta opción seleccionada, el VI comenzará a recolectar datos desde una fecha determinada en el documento de texto date.txt ubicado en C:\Users\USER\Desktop\RT

Microrred\BASE_DATOS\LLBs. En un principio, este fichero contendrá la última fecha en la que se ejecutó el VI con éxito, pero si queremos rescatar datos desde una fecha específica, podemos escribirla en dicho fichero y el programa actuará desde ahí. Esta opción está pensada para ser elegida en caso de que se haya producido algún tipo de error y el PC no haya podido recoger los datos en un tiempo determinado. El VI continúa ejecutándose normalmente.

- Iniciar control y ejecución

Esta opción comienza su ejecución contando con la fecha actual, sin importar qué fecha esté escrita en el fichero. Esta opción servirá en el caso de que no deseemos recoger datos anteriores, bien porque sea la primera vez que ejecutamos este VI o porque, sin ser la primera vez, sabemos que los datos están siendo recogidos con normalidad hasta este momento. El VI continúa ejecutándose normalmente.

- Cancelar

En caso de elegir esta opción se mostrará una ventana emergente como la que se puede ver en la Figura 3.79 y se detendrá la ejecución del VI.

Figura 3.79: Cancelar ejecución VI recolector FTP

c) Diagrama de bloques

En este VI, no hay parte de configuración y todo el VI se trata de la ejecución. Podemos diferenciar la parte del menú inicial de selección (Figura 3.80) y el resto del programa, que se encuentra dentro de un bucle infinito (Figura 3.81). Ambas secciones se encuentran dentro de una estructura de secuencia, de modo que el menú se ejecute antes que el resto del programa. Esto se da porque, en caso de presionar Cancelar en el menú inicial, no queremos que se inicie el resto del bucle.

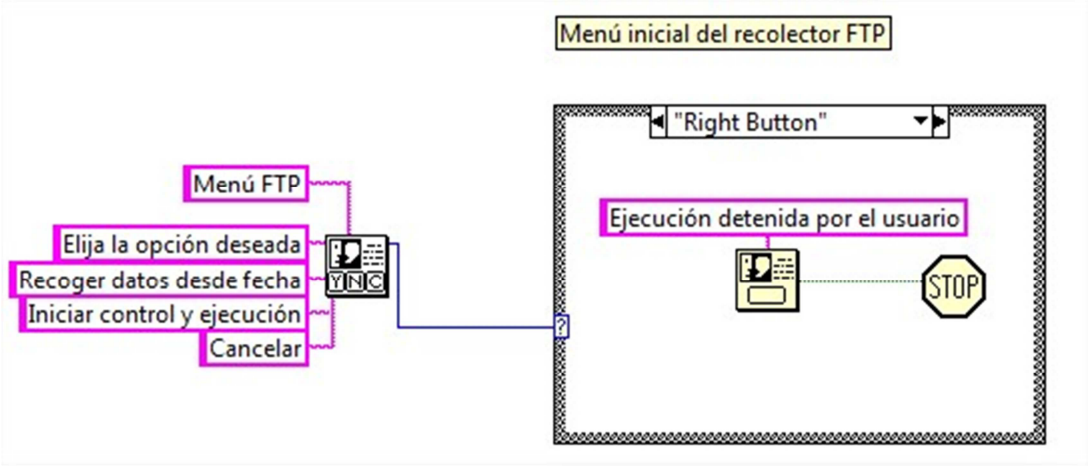

Figura 3.80: Menú inicial recolector FTP

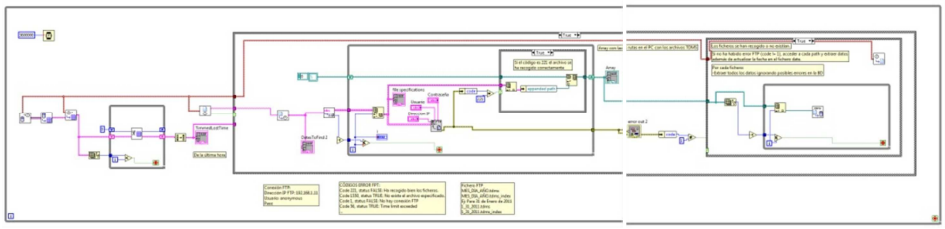

Figura 3.81: Diagrama de bloques del recolector FTP

La parte de ejecución del VI se ejecutará una vez cada hora (3.600.000 milisegundos). En cada iteración realiza varias acciones.

- Comprobación de datos (Figura 3.82)

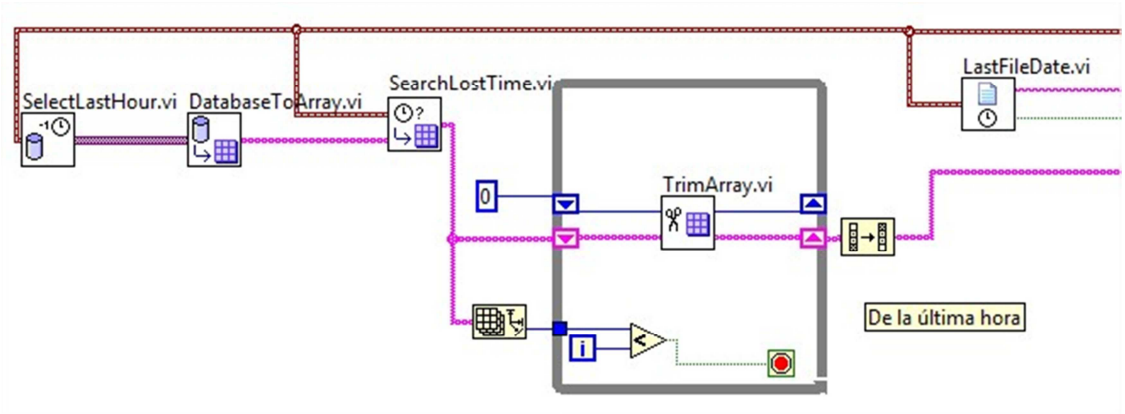

Figura 3.82:Comprobación datos base de datos

Primero hace uso de la función creada SelectLastHour para seleccionar los datos de las fechas y horas registradas en los últimos 60 minutos, es decir, desde la última ejecución de RecolectarFTP. Esos datos son convertidos a un array de cadenas de caracteres mediante la función creada DBToArray. Este array se pasa como argumento a la función creada SearchLostTime de modo, que, contando con el momento en el que se inició la ejecución del VI y contrastando con los datos en el array, obtiene un array con los datos que faltan en esa última hora. Esta estructura será un array de 3600 posiciones, en las que, para cada momento, se mostrará si falta en la base de datos, o aparecerá en blanco en caso contrario. La función de TrimArray es eliminar esos espacios en blancos que indican datos ya existentes. Así pues, tendremos como resultado, una vez que operemos con reverse, un array de los momentos de la última hora que faltan en la base de datos.

Por otro lado, la función LastFileDate nos devolverá la fecha del fichero de texto antes mencionado. También devolverá un booleano que valdrá TRUE en caso de que la fecha del fichero no sea la hora inmediatamente anterior a la actual.

- El resto de la ejecución sigue el siguiente algoritmo:
  - Si fecha_fichero = hora_anterior (LastFileDate devuelve FALSE)
    - Si no faltan datos de la última hora (tamaño del array con horas que faltan = 0)
      - Cambiar fecha en el fichero a actual. (SetTimeFile.vi)
    - Si faltan datos (tamaño array con momentos que faltan != 0) (Figura 3.83)

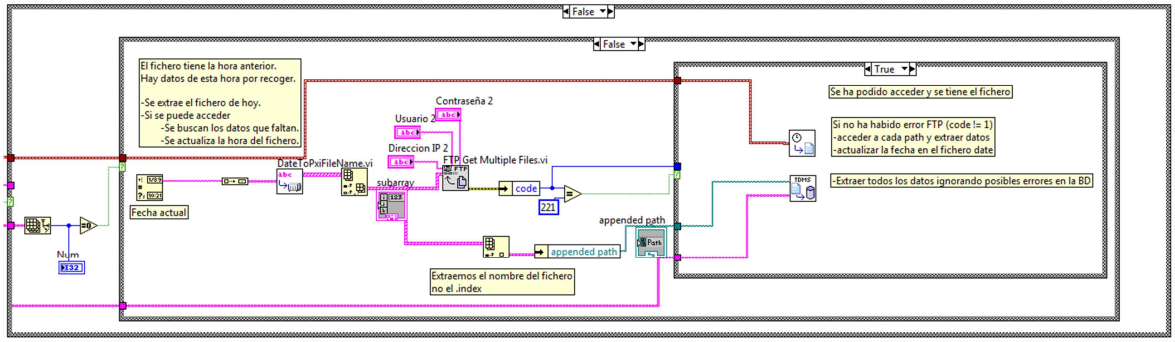

Figura 3.83: Código si recoge únicamente datos de la última hora

- Recoger fichero último día y añadir únicamente los datos que faltan
- Cambiar fecha en el fichero actual (SetTimeFile.vi)

  - Si fecha_fichero != hora_anterior (LastFileDate devuelve TRUE) (Figura 3.84)

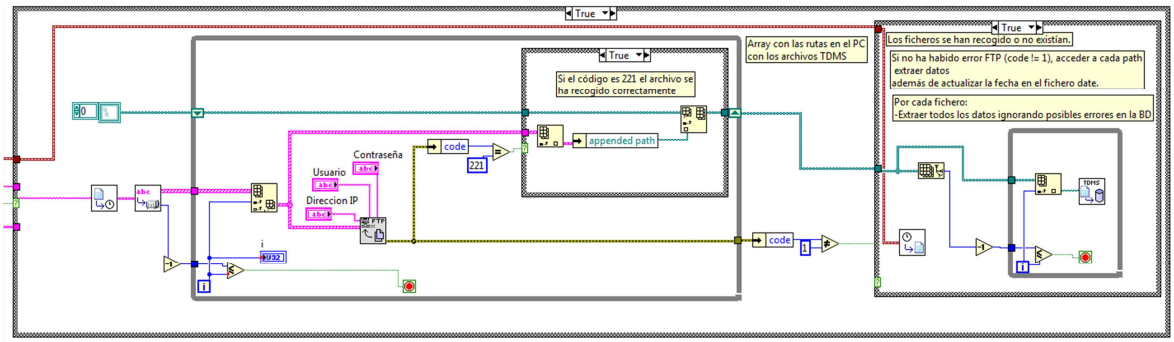

Figura 3.84: Código si recoge datos de varios días

- Generar fechas que faltan (a partir de fecha_fichero) (ExtractDates.vi)
- Generar nombres ficheros a recoger (DateToPxiFileName.vi)
- Recoger ficheros de esas fechas (FTP Get Multiples Files)
- Insertar todos los datos (TDMStoDatabase.vi)
- Cambiar hora en el fichero actual (SetTimeFile.vi)

La conexión FTP se establece con la dirección IP 192.168.1.11, usuario anonymous y sin contraseña.

A la hora de extraer los ficheros, si los mismos no existen en el PXi se considerará que se han recogidos y se actualizará la hora del fichero con la fecha. Si no se han podido extraer debido a que hay error de conexión FTP entre el PC y el PXi, no se actualiza la hora y en la próxima ejecución se volverán a intentar rescatar todos los ficheros.

Los códigos más comunes que devuelve la función que recolecta ficheros FTP son:

- Code 221, status FALSE: Ha recogido bien los ficheros.
- Code 1550, status TRUE: No existe el archivo especificado.
- Code 1, status FALSE: No hay conexión FTP
- Code 56, status TRUE: Time limit exceeded

d) Jerarquía del VI

Los SubVIs se pueden ver en la Figura 3.85. Para consultar su funcionamiento interno, dirigirse a la sección 3.2.10.

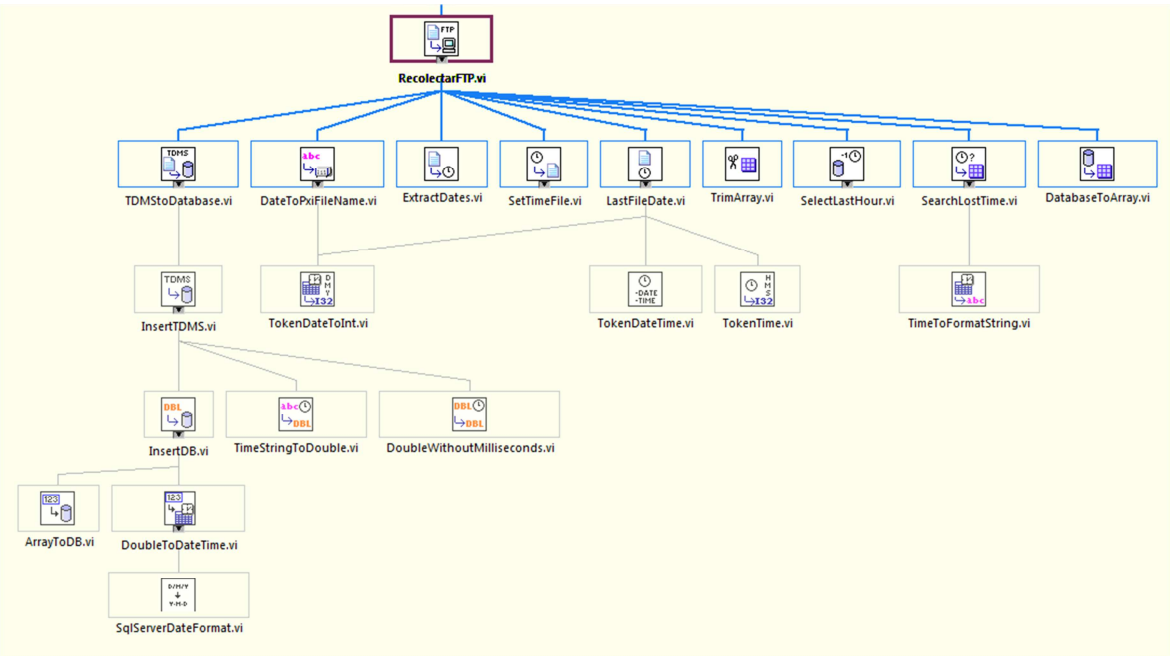

Figura 3.85: SubVIs utilizados

SubVIs utilizados:

- TDMStoDatabase
  - InsertTDMS
    - InsertDB
      - ArrayToDB
      - DoubleToDateTime
        - SqlServerDateFormat
    - TimeStringToDouble
    - DoubleWithoutMilliseconds
- DateToPxiFilename
  - TokenDateToInt
- ExtractDates
- SetTimeFile
- LastFileDate
  - TokenDateToInt
  - TokenDateTime
  - TokenTime
- TrimArray
- SelectLastHour
- SearchLostTime
  - TimeToFormatString
- DatabaseToArray

3.2.9. Interacción con Base de Datos

a) Descripción

El programa de interacción con la Base de Datos (MenuDB.vi) aporta un interfaz amigable para la interacción sencilla con la Base de Datos, permitiendo realizar las operaciones básicas con ella, de modo que no se requieran conocimientos sobre la estructura interna de la base de datos para modificarla.

Se presupone que el futuro usuario de la aplicación no tiene los conocimientos necesarios sobre Bases de Datos para interactuar con el interfaz de SQL Server. Pero, así como para facilitar la tarea al usuario, también se ha diseñado este VI para proporcionar protección a la base de datos, ya que interactuar directamente desde el interfaz de SQL Server sin estar completamente seguro puede resultar en un desastre para la base de datos (de todos modos, en ese caso, se podría restaurar la base de datos a una versión correcta, sección 4.3.5). Este VI proporciona funcionalidades básicas que se ejecutan de manera controlada y sin ningún riesgo para la Base de Datos. Se entiende que no se ejecutará periódicamente, ya que los cambios en la estructura de la Microrred serán mínimos.

b) Panel Frontal

Icono: 

El icono representa un menú y una base de datos.

Se puede observar en la Figura 3.86 que el interfaz con el usuario es muy sencillo.

Figura 3.86: MenuDB: panel frontal

Las diferentes opciones del menú son:

- Información Base de Datos (Figura 3.87)

  Devuelve la información correspondiente a la Base de Datos, especificando la relación, columnas, tipo de datos y tamaño de los mismos. Pulsando en el botón SALIR se vuelve al menú de inicio.

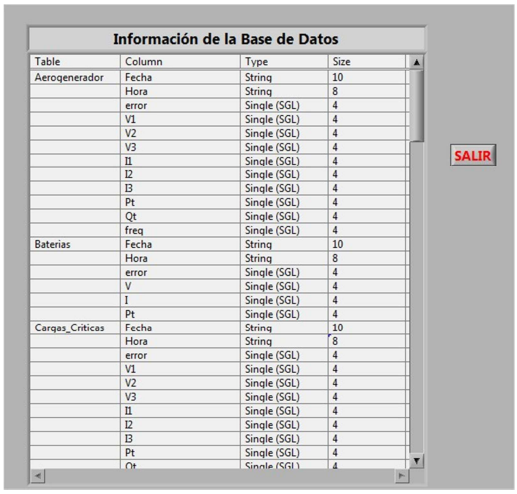

Figura 3.87: Información Base de Datos

- Añadir módulo (Figura 3.88)

Añade un nuevo módulo o dispositivo a la Base de Datos. Se especifica el nombre del nuevo módulo y su lista de atributos con los tipos de datos correspondientes. Pulsando en el botón GO aparece una ventana emergente para confirmar la operación (Figura 3.89). En caso de pulsar Cancelar no se ejecutará ninguna operación y se volverá al menú inicial. Pulsando en el botón SALIR se vuelve al menú anterior sin realizar ningún cambio. Si creamos un nuevo módulo, la ventana se cerrará y volverá al menú anterior una vez haya finalizado las operaciones.

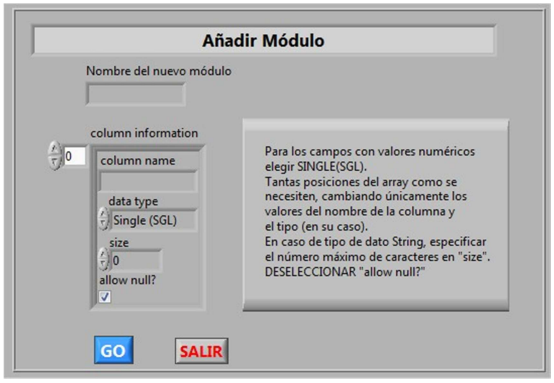

Figura 3.88: Añadir Módulo

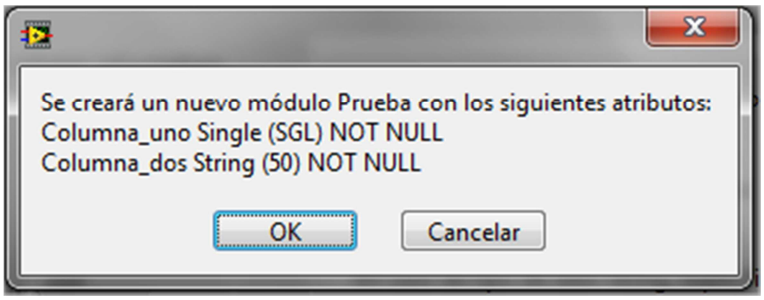

Figura 3.89: Confirmar añadir módulo

- Eliminar módulo (Figura 3.90)

Elimina uno de los módulos existentes de la Base de Datos. Se elige uno de los módulos del Menú y se pulsa en GO para eliminar. Aparecerá una ventana emergente para confirmar la operación. (Figura 3.91) En caso de pulsar Cancelar no se ejecutará ninguna operación y se volverá al menú inicial. Pulsando en el botón SALIR se vuelve al menú anterior sin realizar ningún cambio. Si eliminamos algún módulo, la ventana se cerrará y volverá al menú anterior una vez haya finalizado las operaciones.

Figura 3.90: Eliminar módulo

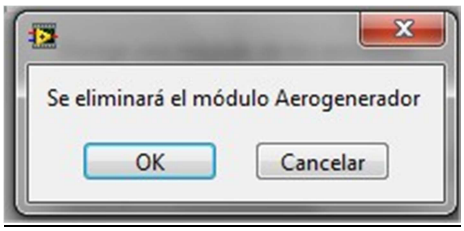

Figura 3.91: Confirmar eliminar módulo

- Añadir atributo (Figura 3.92)

Añade un atributo a un módulo o dispositivo ya existente en la Base de Datos. Seleccionar un módulo ya existente y establecer el nombre y tipo del nuevo atributo. Pulsando GO aparecerá una ventana emergente (Figura 3.93) para confirmar la operación y pulsando SALIR se volverá al menú inicial. Si añadimos algún atributo, la ventana se cerrará y volverá al menú anterior una vez haya finalizado las operaciones.

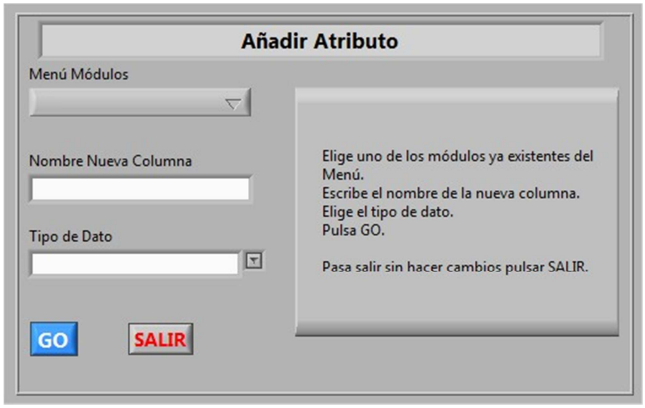

Figura 3.92: Añadir atributo

Figura 3.93: Confirmar añadir atributo

- Eliminar atributo(Figura 3.94)

Se elimina un atributo de un módulo o dispositivo ya existente en la Base de Datos. Seleccionar el módulo, pulsar en “Consultar atributos” y elegir del menú “Atributo a eliminar”. Pulsando GO aparecerá una ventana emergente (Figura 3.95) para confirmar la operación y pulsando SALIR se volverá al menú inicial. Si eliminamos algún atributo, la ventana se cerrará y volverá al menú anterior una vez haya finalizado las operaciones.

No se muestran para eliminar los atributos Fecha y Hora, ya que ambos pertenecen a la clave primaria de la relación, por lo que no es posible eliminarlos.

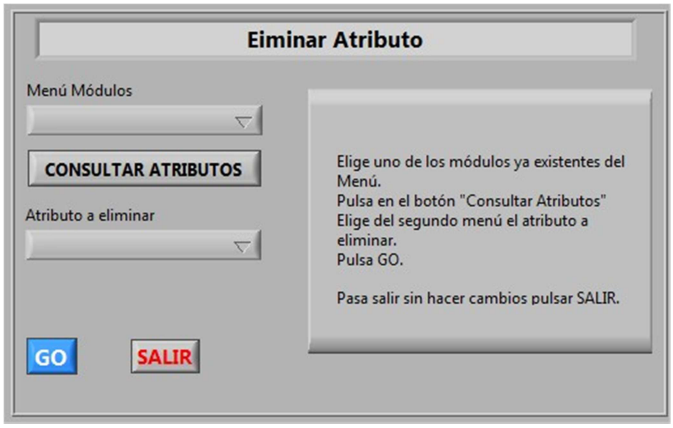

Figura 3.94: Eliminar atributo

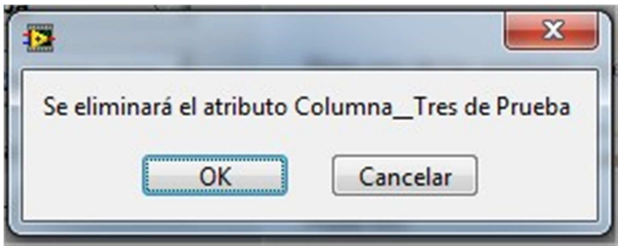

Figura 3.95: Confirmar eliminar atributo

c) Diagrama de bloques

El diagrama de bloques se reduce a lo mostrado en la Figura 3.96.

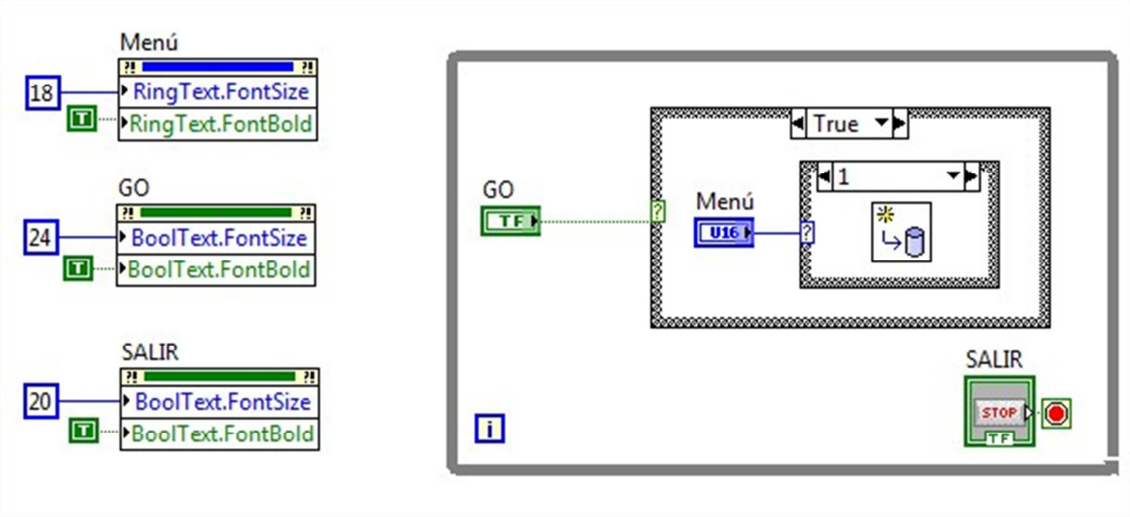

Figura 3.96: MenuDB: diagrama de bloques

Únicamente cuenta con la configuración de los botones y el menú y un bucle infinito. Al pulsar el botón GO, se comprueba qué opción se ha elegido en el menú y se ejecuta el SubVI correspondiente.

Para que los SubVIs aparezcan como ventanas emergentes, hay que acceder a las propiedades del VI (File/Vi Properties) y en la sección Window Appearance pulsar en Customize y seleccionar:

-Show Front Panel when called
-Close afterwards if originally closed

Las acciones son llevadas a cabos por las siguientes funciones creadas:

- Información Base de Datos: GetDatabaseInformation
- Añadir módulo: CreateTable
- Eliminar módulo: DropTable
- Añadir atributo: AddColumn
- Eliminar atributo: DropColumn

d) Jerarquía del VI

Los SubVIs se pueden ver en la Figura 3.97. Para consultar su funcionamiento interno, dirigirse a la sección 3.2.10.

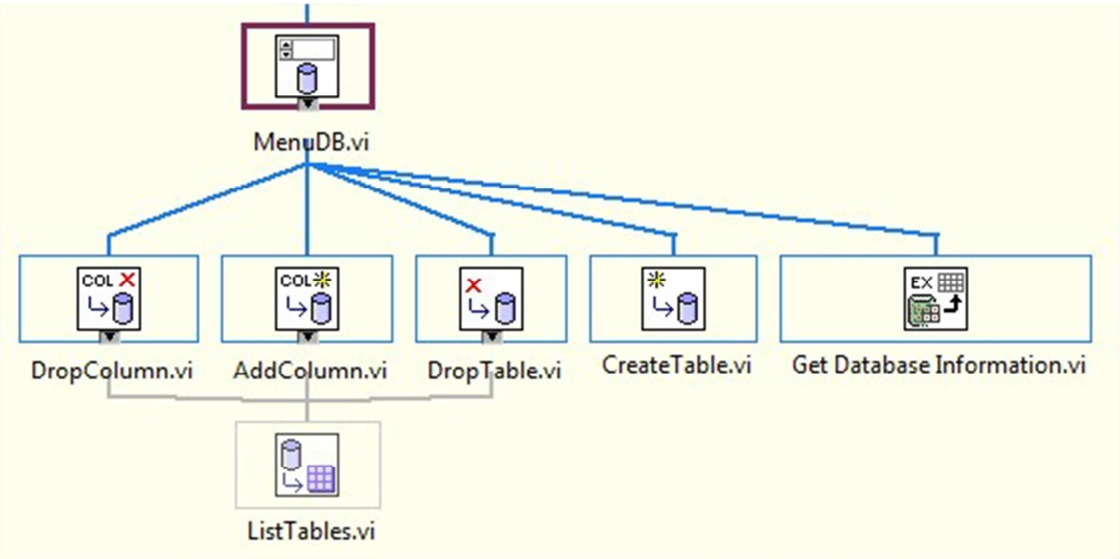

Figura 3.97: SubVIs utilizados

SubVIs utilizados:

- DropColumn
  - ListTables
- AddColumn
  - ListTables
- DropTable
  - ListTables
- CreateTable
- GetDatabaseInformation

3.2.10. Librerías creadas

Se han creado y utilizado un total de 39 SubVIs que se han agrupado en librerías según su tipo para contribuir a la correcta organización del proyecto y a la identificación de los SubVis.

Además de las librerías, se incluye en este apartado la descripción de Global.vi, que, si bien no está alojado en una librería, es un VI que toma partido de forma fundamental en el proyecto.

Para cada librería se especificará su descripción y luego, para cada uno de los SubVIs que la componen, su definición, sus argumentos de entrada y salida, funcionamiento y ejemplo de funcionamiento, si procede.

El índice de librerías con sus correspondientes SubVIs es el siguiente:

a) Global

b) SQLToolsLLB
   i. FROMclause
   ii. ORDERBYclause
   iii. SELECTclause
   iv. WHEREclause
   v. WHEREerrorClause

c) ArrayToolsLLB
   i. ExtractDates
   ii. Media
   iii. SearchLostTime
   iv. SelectedArray
   v. SelectedTables
   vi. TrimArray

d) DBToolsLLB
   i. AddColumn
   ii. ArrayToDB
   iii. CompleteArray

iv. CreateTable

v. DBToArray

vi. DropColumn

vii. DropTable

viii. Get Database Information

ix. InsertDB

x. InsertTDMS

xi. ListTables

xii. SelectLastHour

xiii. SqlServerDateFormat

xiv. TDMStoDatabase

xv. xToArray

xvi. xToString

e) FileToolsLLB

i. CreateFilename

ii. DateToPxiFileName

iii. LastFileDate

iv. SetTimeFile

v. WriteToXLS

f) ChartToolsLLB

i. CheckError

ii. Draw Picture

iii. MenuToChart

g) StringAndNumericToolsLLB

i. ArrayToString

ii. DoubleToDateTime

iii. DoubleWithoutMilliseconds

iv. StringToTimestamp

v. TimeStringToDouble

vi. TimeToFormatString

vii. TokenDateTime

viii. TokenDateToInt

ix. TokenTime

x. xToString

a) Global

El VI Global (Global.vi) contiene las variables globales utilizadas a lo largo del proyecto. LabVIEW organiza las variables globales en un .vi especial que únicamente consta de panel frontal, y no de diagrama de bloques.

El panel frontal correspondiente a Global.vi se puede ver en la Figura 3.98.

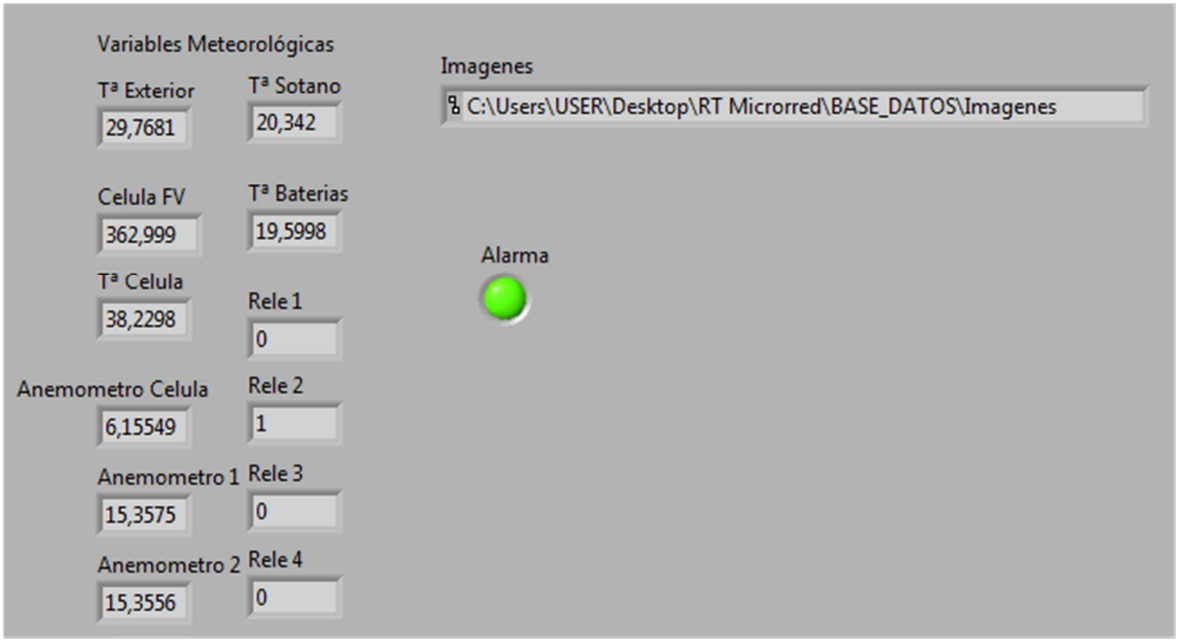

Figura 3.98: Panel frontal de las variables globales

Las variables globales utilizadas en el proyecto son las correspondientes a las variables meteorológicas, al path de la carpeta con las imágenes utilizadas y a un booleano de alarma que se apagará si la comunicación entre el Pxi y el PC falla.

El path es inicializado en el propio VI, mientras que las variables meteorológicas son asignadas desde las gráficas en tiempo real. Así, tanto el Esquema General como el Esquema General Térmico pueden acceder a estas variables sin tener que volver a extraerlas del array. Con esto, se consigue una mejora en la eficiencia de la aplicación.

Si bien es cierto que, si no se ejecutan las gráficas en tiempo real, no se podrán ver los datos en los Esquemas Generales. Así pues, si se desea desligar esta dependencia, habría que utilizar la misma técnica que en las gráficas en tiempo real para extraer todos los datos de las variables meteorológicas.

El valor de Alarma se ha obtenido de la variable del mismo nombre de TCP Communication - Host.vi, que pertenece al mismo proyecto de LabVIEW pero no entra dentro de mi aplicación.

Icono: 
Es el icono por defecto definido por LabVIEW para las variables globales.

b) SQLToolsLLB

Agrupa las librerías que generan consultas que se ejecutarán sobre SQL Server. Así, pueden ser reutilizadas en cualquier aplicación que implique la generación de consultas SQL de dificultad media o alta.

SubVIs pertenecientes a la librería:

i. SELECTclause

El SubVI SELECTclause.vi genera la cláusula SELECT de una sentencia SQL para ser ejecutada sobre SQLServer. Establecerá los distintos atributos sobre los que se extraerán datos.

Icono: 

Representa la cláusula SELECT que se ejecuta sobre una Base de Datos.

Argumentos de entrada:
-Selected Tables: Array con los nombres de las tablas seleccionadas
-Var Array: Array con los nombres de las variables seleccionadas

Argumentos de salida:
-SELECT Clause: Cadena de caracteres con la cláusula SELECT

El funcionamiento del subVI se puede ver en la Figura 3.99.

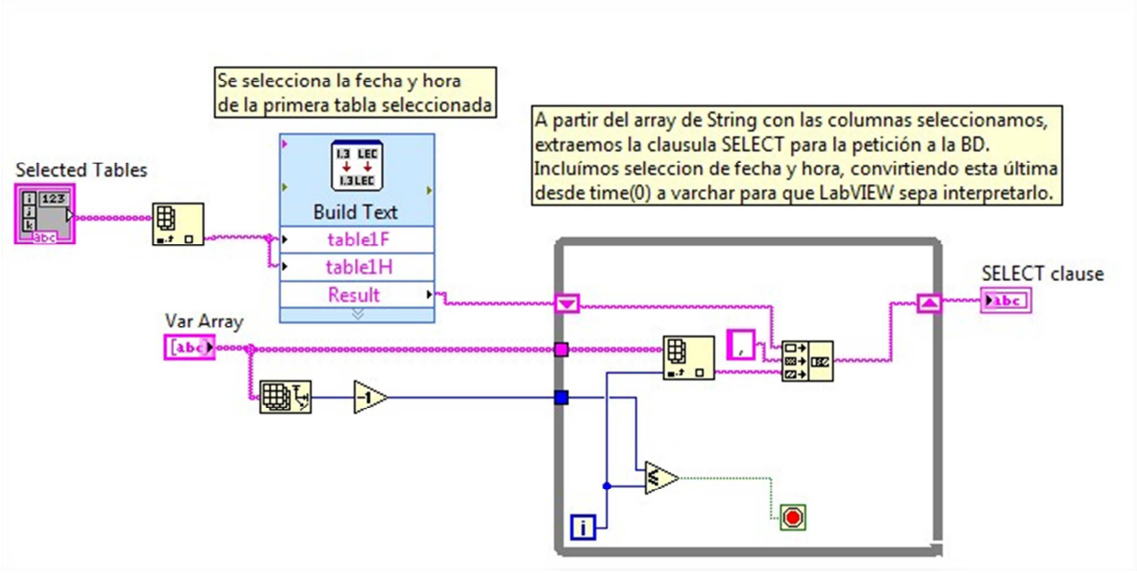

Figura 3.99: SELECTclause: diagrama de bloques

BuildText genera lo siguiente:

SELECT tabla1.Fecha, CONVERT (varchar, tabla1, Hora, 8) Hora

Es decir, al principio de la cláusula SELECT, siempre se seleccionará la Fecha y Hora. La hora está almacenada en SQL Server como tipo Time (0), pero LabVIEW no lo reconoce, así que al hacer la selección se convierte a cadena de caracteres y se renombra como Hora.

Puede que estemos seleccionando datos de más de un dispositivo, por lo que seleccionar solamente “Fecha” conllevaría una ambigüedad, ya que todos los dispositivos contienen este campo. De todas formas, todas las Fechas y Horas contienen los mismos datos, así que no importa sobre qué relación en concreto los seleccionemos. Así, seleccionamos el primero de los valores de la variable SelectedTables, que corresponderá a una de las tablas de las que se selecciona los datos, y se lleva a BuildText, de forma que construya la parte de la sentencia correcta.

Después se ejecuta un bucle para cada una de las posiciones del array con las variables seleccionadas. Simplemente, se concatenará a la cláusula generada hasta ahora la variable actual, separada por una coma.

Podemos ver un ejemplo de funcionamiento en la Figura 3.100.

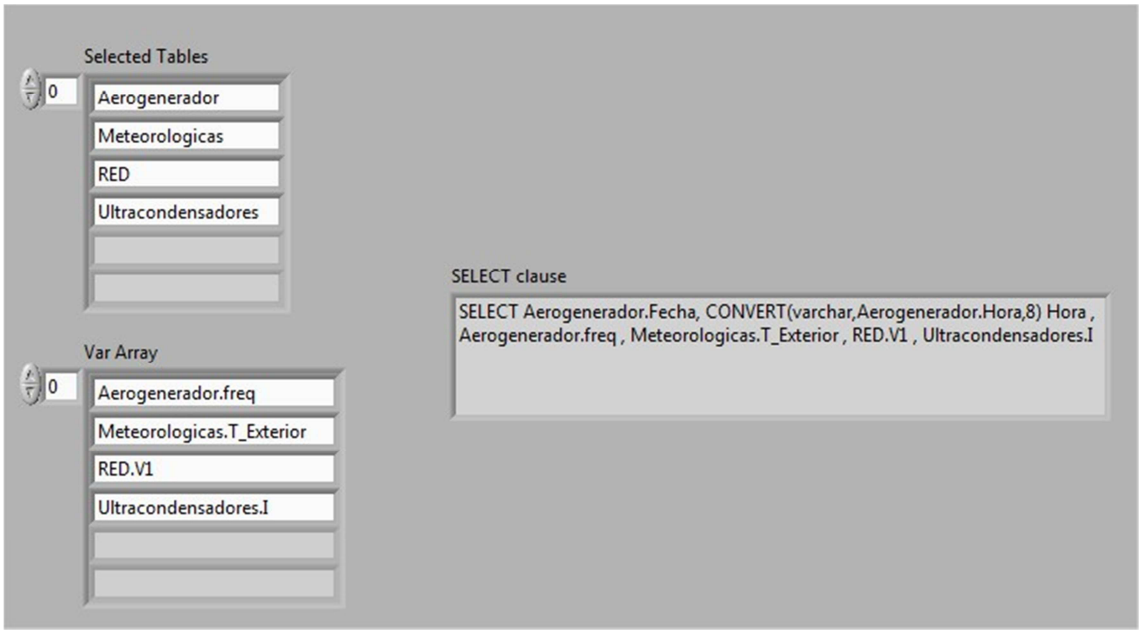

Figura 3.100: Funcionamiento SELECTclause

ii. WHEREclause

El SubVI WHEREclause.vi genera parte de la cláusula WHERE de una sentencia SQL para ser ejecutada sobre SQLServer. Corresponderá a la parte que restringe las fechas de los datos.

Icono: 

Representa la cláusula WHERE que se ejecuta sobre una Base de Datos dependiendo de la opción para representar.

Argumentos de entrada:

-Menú Dates: Opción del Menú
-Fecha Inicio: Primera fecha desde la que se recogen datos
-Hora Inicio: Primera hora desde la que se recogen datos
-Fecha Fin: Última fecha hasta la que se recogen datos
-Hora Fin: Última hora hasta la que se recogen datos
-SelectedTables: array con las tablas seleccionadas

Argumentos de salida:

-Where clause: cadena de caracteres con la cláusula WHERE

El funcionamiento del subVI se puede ver en la Figura 3.101.

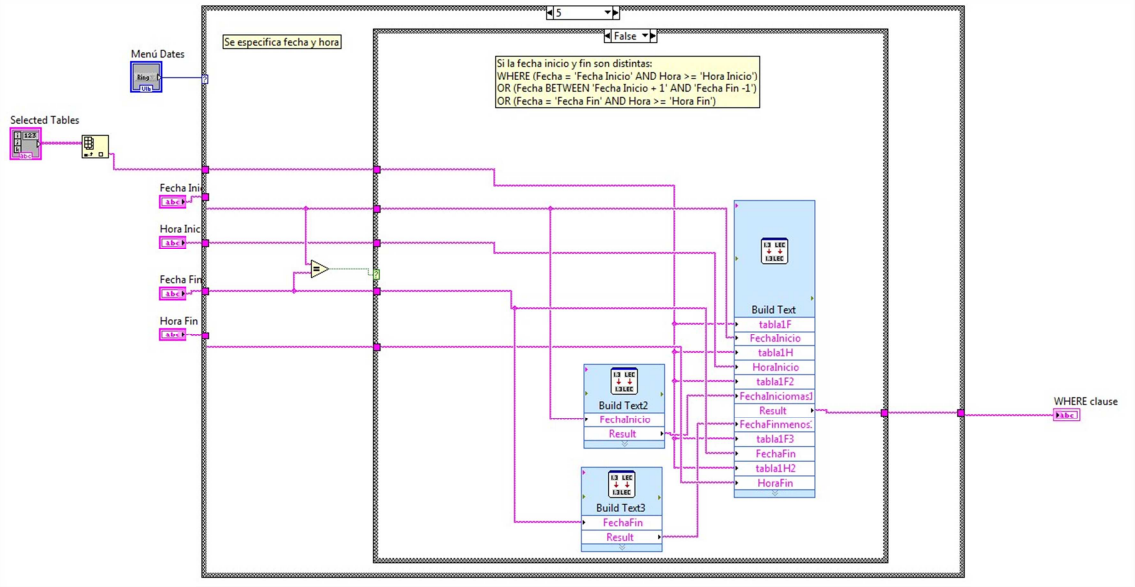

Figura 3.101: WHEREclause: diagrama de bloques

El VI consta de una estructura CASE principal que dependerá del valor de MenúDates. En todos los casos, tabla1 es el valor de la primera de las tablas en SelectedTables. Según el valor del menú:

- 0: seleccionará los datos de la última hora, generando una cláusula como la siguiente:

```
WHERE tabla1.Fecha = convert(date,getdate())
AND tabla1.Hora > convert (varchar, dateadd(hour,-1,getdate()),8)
```

Es decir, la fecha es la misma que la actual, y la hora, todas las existentes desde hace 1 hora hasta el momento actual.

- 1: seleccionará los datos del último día (últimas 24h), generando una cláusula como la siguiente:

```
WHERE ((tabla1.Fecha = (convert(date,getdate()-1))
        AND tabla1.Hora > (convert(varchar, getdate(),8)) )
OR (taba1.Fecha > (convert(date,(getdate() -1)))))
```

Es decir, la fecha es la actual restándole 1 día y la Hora mayor que la actual o la fecha es igual a la actual. Entonces, si en el momento de la consulta son las 12:34:56h, los datos serán desde las 12:34:56h del día anterior hasta las 12:34:56h del día actual.

- 2:seleccionará los datos de la última semana (últimos 7 días), generando una cláusula como la siguiente:

```
WHERE ((tabla1.Fecha = (convert(date,(getdate()-7) ))
        AND tabla1.Hora > (convert(varchar, getdate(),8)) )
OR (tabla1F.Fecha > (convert(date,getdate() -7))))
```

Es decir, la fecha actual restándole 7 días y la hora mayor que la actual o la fecha mayor que la actual restándole 7 días. Entonces, si en el momento de la consulta son las 12:34:56h de un domingo, los datos serán desde las 12:34:56h del lunes de esa semana hasta las 12:34:56h del día actual.

- 3: seleccionará los datos del último mes (últimos 30 días), generando una cláusula como la siguiente:

```
WHERE ((tabla1.Fecha = (convert(date,getdate()-30) )
        AND tabla1.Hora > (convert(varchar, getdate(),8)) )
    OR (tabla1.Fecha > (convert(date,getdate() -30))))
```

Es decir, desde la fecha actual restándole 30 días y la hora mayor que la actual o la fecha mayor que la actual restándole 30 días. Entonces, si en el momento de la consulta son las 12:34:56h de un día 30, los datos serán desde las 12:34:56h del día 1 de ese mismo mes hasta las 12:34:56h del día actual.

- 4: seleccionará los datos del último año (últimos 365 días), generando una cláusula como la siguiente:

```
WHERE ((tabla1.Fecha = (convert(date,getdate()-365) )
        AND tabla1.Hora > (convert(varchar, getdate(),8)) )
    OR (tabla1.Fecha > (convert(date,getdate() -365))))
```

Es decir, desde la fecha actual restándole 365 días y la hora mayor que la actual o la fecha mayor que la actual restándole 365 días. Entonces, si en el momento de la consulta son las 12:34:56h del día 31 de Diciembre, los datos serán desde las 12:34:56h del día 1 de Enero de ese mismo año.

- 5: seleccionará los datos comprendidos entre las fechas y horas especificadas. En caso de que las fechas coincidan en un mismo día, es decir, si solamente hemos especificado un intervalo en un mismo día, la cláusula generada es como la siguiente:

  ```
  WHERE table1.Fecha='FechaInicio'
  AND tabla1.Hora BETWEEN 'HoraInicio' AND 'HoraFin'
  ```

Es decir, con la fecha especificada entre la hora de inicio y la de fin.

En cambio, si la fecha no coincide, la cláusula generada es como la siguiente:

```
WHERE (tabla1.Fecha = 'FechaInicio' AND tabla1.Hora>= 'HoraInicio')
OR (tabla1.Fecha BETWEEN DATEADD (day,+1,'FechaInicio') AND DATEADD (day,-1,'FechaFin'))
OR (tabla1.Fecha= 'FechaFin' AND tabla1.Hora <= 'HoraInicio')
```

Es decir, que coincida con la fecha de inicio y sea mayor que la hora actual, que esté entre el día siguiente del día de inicio y el día anterior al día final o que la fecha coincida con la fecha de fin y la hora sea menor que la hora de fin.

Podemos ver un ejemplo de funcionamiento en la Figura 3.102.

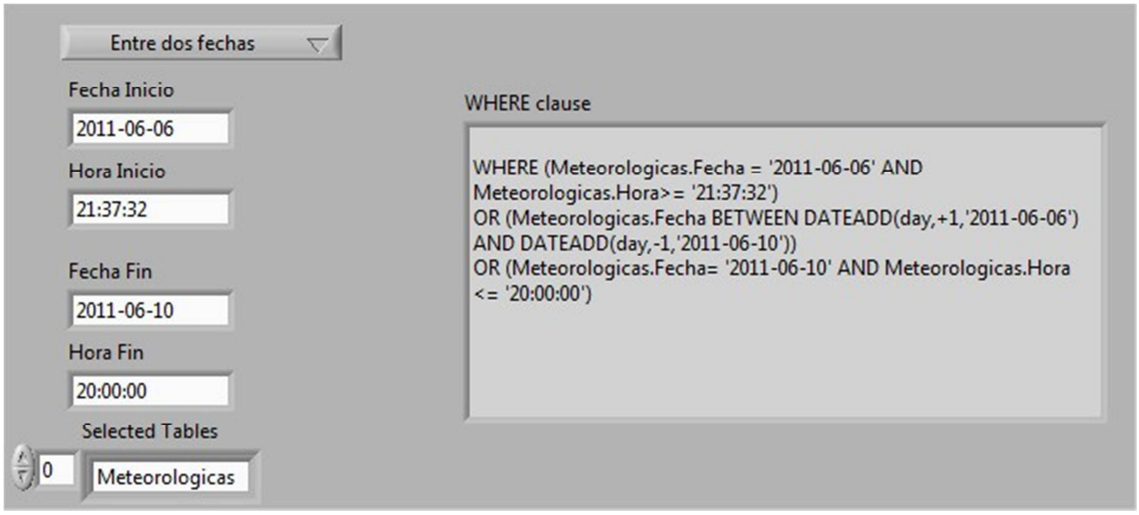

Figura 3.102: Funcionamiento de WHEREclause

iii. WHEREerrorClause

El SubVI WHEREclause.vi genera parte de la cláusula WHERE de una sentencia SQL para ser ejecutada sobre SQLServer. Corresponderá a la parte que restringe los datos a los no erróneos.

Icono: 

Representa la cláusula WHERE que se ejecuta sobre una Base de Datos comprobando los errores.

Argumentos de entrada:
-Table: tabla actual
-Col Selected: array con las columnas seleccionadas de esa tabla

Argumentos de salida:
-WHERE error: cadena de caracteres con la cláusula WHERE correspondiente al error

El funcionamiento se del subVI se puede ver en la Figura 3.103.

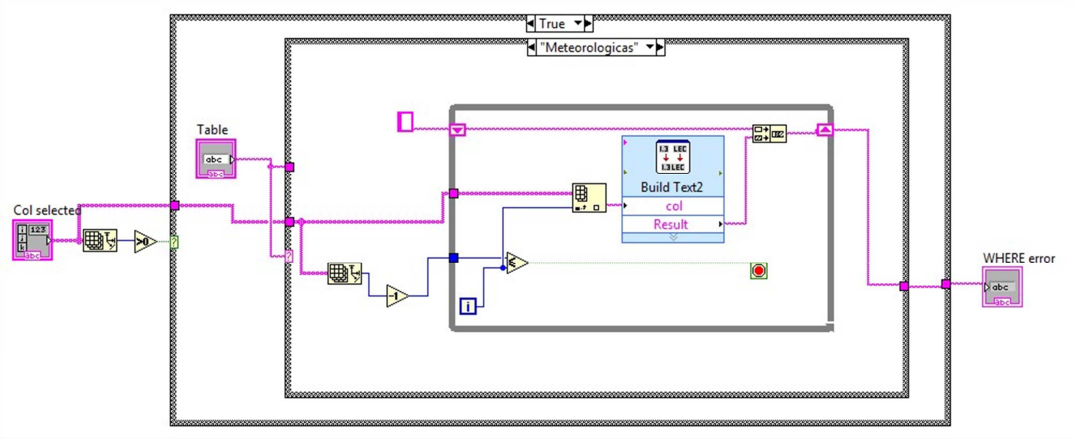

Figura 3.103: diagrama de bloques de WHEREerrorClause

El subVI contiene dos estructuras CASE anidadas. La primera de ellas comprueba el número de columnas que se han seleccionado de esa tabla. Si se ha elegido alguna (size > 0) entonces se ejecutará el código interno. Si no, no tiene sentido restringir la consulta a que esa tabla no contenga errores, dado que no deseamos ningún valor de ella.

En caso de que hayamos elegido alguna variable, la segunda estructura CASE actúa diferente dependiendo de la tabla que estemos tratando.

Si estamos en las variables meteorológicas, el error en los datos se marca con el valor -100. Entonces, para cada una de las variables seleccionadas añadiremos a la consulta lo siguiente:

AND column <> -100

En caso de encontrarnos en cualquier de las otras tablas, la cláusula generada será la siguiente:

AND table.error = 0

Lo que quiere decir que la variable de error de la tabla no valga distinto de 0.

Así, solo seleccionaremos los datos sin errores.

Podemos ver un ejemplo de funcionamiento en la Figura 3.104.

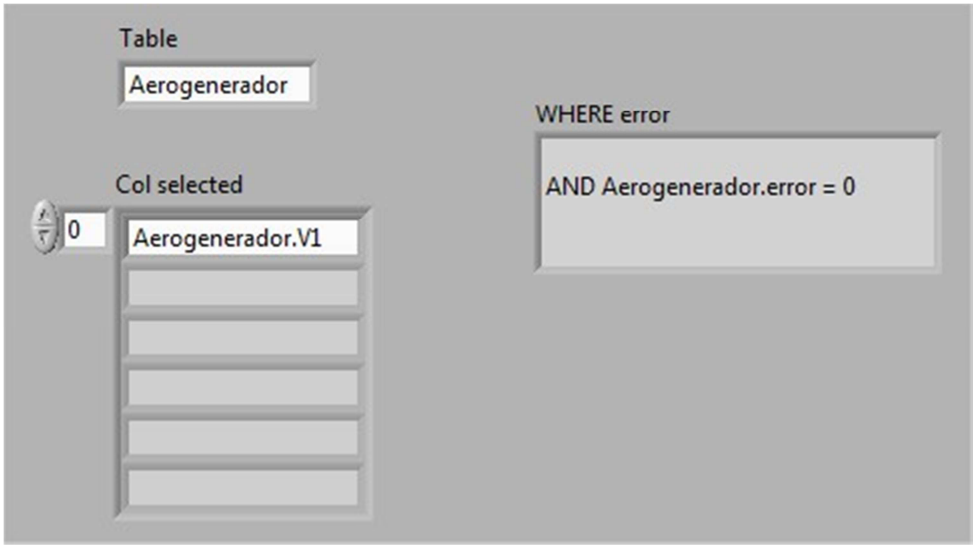

Figura 3.104: Funcionamiento de WHEREerrorClause

iv. FROMclause

El SubVI FROMclause.vi genera la parte de la cláusula FROM de una sentencia SQL para ser ejecutada sobre SQLServer. Esta cláusula selecciona de qué dispositivos se extraerán los datos, y establece las relaciones necesarias entre ellos para que los datos sean los correctos.

Icono: 

Representa la cláusula FROM que se ejecuta sobre una Base de Datos.

Argumentos de entrada:
-SelectedTables: array con los distintos dispositivos de los que se ha seleccionado alguna variable

Argumentos de salida:
-FROM clause: cadena de caracteres con la cláusula FROM

El funcionamiento del SubVI se puede ver en la Figura 3.105.

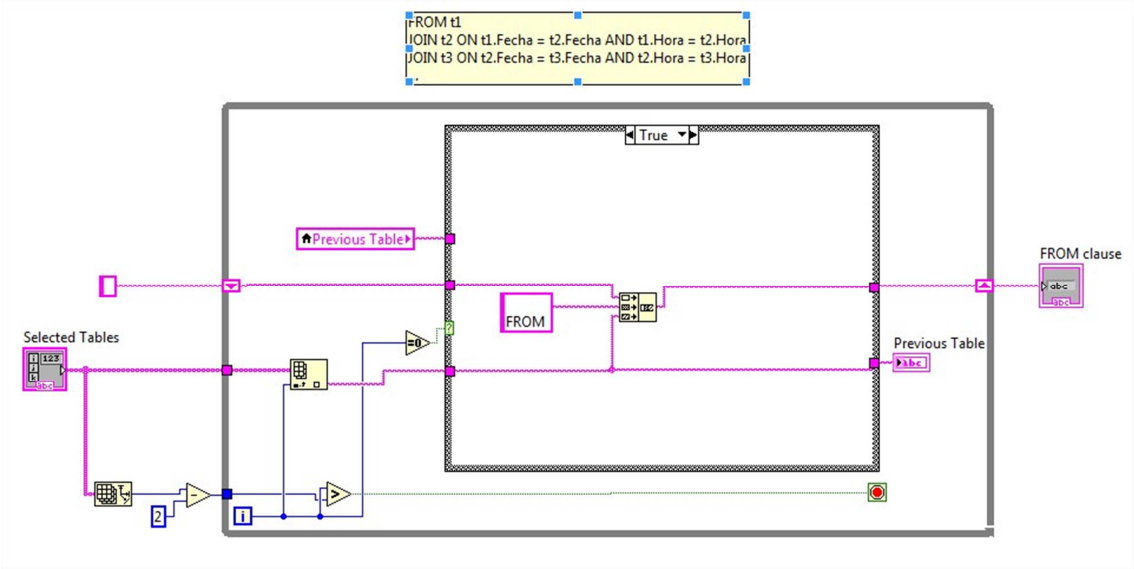

Figura 3.105: Diagrama de bloques de FROMclause.vi

La cláusula FROM a generar sigue la siguiente estructura:

FROM tabla1
JOIN tabla2 ON tabla1.Fecha = tabla2.Fecha AND tabla1.Hora = tabla2.Hora
JOIN tabla3 ON tabla2.Fecha = tabla3.Fecha AND tabla2.Hora = tabla3.Hora
…

Es decir, se selecciona de la primera tabla y luego se van uniendo las demás, de modo que la hora y fecha de la primera coincida con la hora y fecha de la segunda. De manera gráfica, sería algo similar a lo representado en la Figura 3.106.

Figura 3.106: Representación gráfica de cláusula FROM

En el código, para la primera tabla en SelectedTables se genera la parte :
FROM tabla1
En todo momento se almacena en la variable PreviousTable la última tabla comprobada, así que en la siguiente iteración, correspondiente a la segunda de las tablas, se unirá con la primera añadiendo:
JOIN tabla2 ON tabla1.Fecha = tabla2.Fecha AND tabla1.Hora = tabla2.Hora

Podemos ver un ejemplo de funcionamiento en la Figura 3.107.

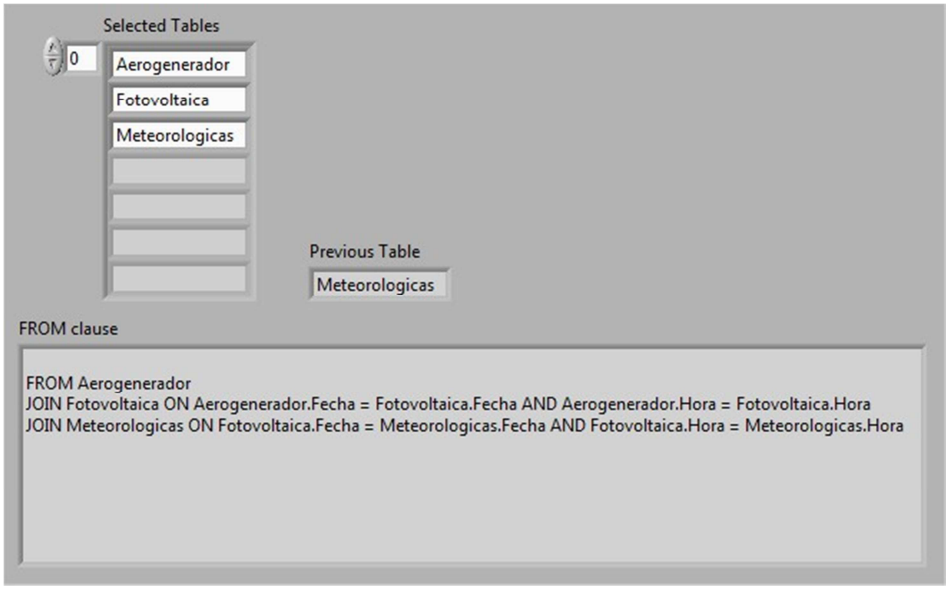

Figura 3.107: Funcionamiento de FROMclause

v. ORDERBYclause

El SubVI ORDERBYclause.vi genera la cláusula ORDERBY de una sentencia SQL para ser ejecutada sobre SQLServer. Esta cláusula ordena los datos en orden cronológico.

Icono: 

Representa la cláusula ORDERBY que se ejecuta sobre una Base de Datos.

Argumentos de entrada:
-Selected Tables: array con los nombres de los distintos dispositivos seleccionados

Argumentos de salida:
-Resultado: cadena de caracteres con la cláusula ORDERBY correctamente construida.

El funcionamiento del SubVI se puede ver en la Figura 3.108.

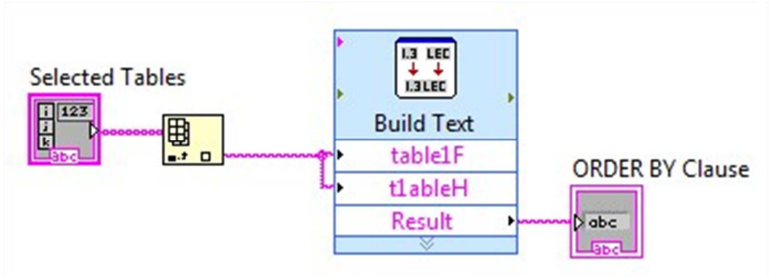

Figura 3.108: Diagrama de bloques del SubVI ORDERBYclause

Lo único que hace es extraer el nombre del primero de los dispositivos del que seleccionamos algo. Con ese nombre, BuildText construye una cláusula como la siguiente:

ORDER BY tabla1.Fecha, tabla1.Hora

Podemos ver un ejemplo de funcionamiento en la Figura 3.109.

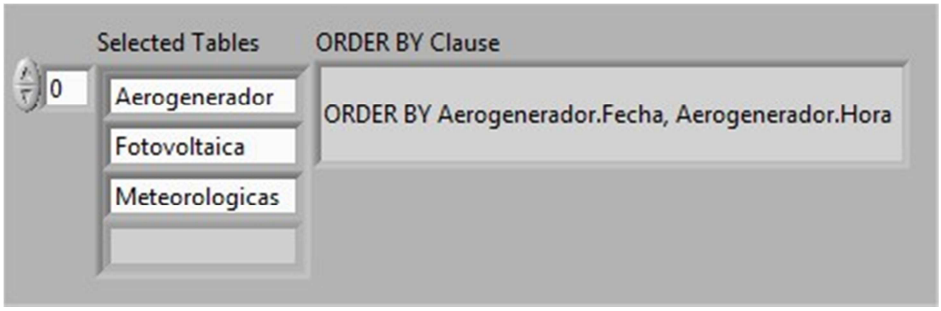

Figura 3.109: Ejemplo de funcionamiento de ORDERBYclause.vi

c) ArrayToolsLLB

Agrupa los SubVIs que actúan sobre arrays, ya sean de dobles o de cadenas de caracteres. Aunque trabajen sobre un mismo tipo de dato, las funcionalidades de los SubVIs difieren mucho.

SubVIs pertenecientes a la librería:

i. ExtractDates

El SubVI ExtractDates devuelve un array con las fechas que han transcurrido desde una determinada.

Icono: 

Representa la salida de un fichero, a partir del cual se generarán días.

Argumentos de entrada:
-LastDate (dd/mm/yyyy): cadena de caracteres con una fecha en formato dd/mm/yyyy

Argumentos de salida:
-Dates: array con las fechas que han transcurrido desde la fecha de entrada hasta la actual, incluyéndolas.

El funcionamiento del SubVI se puede ver en la Figura 3.110.

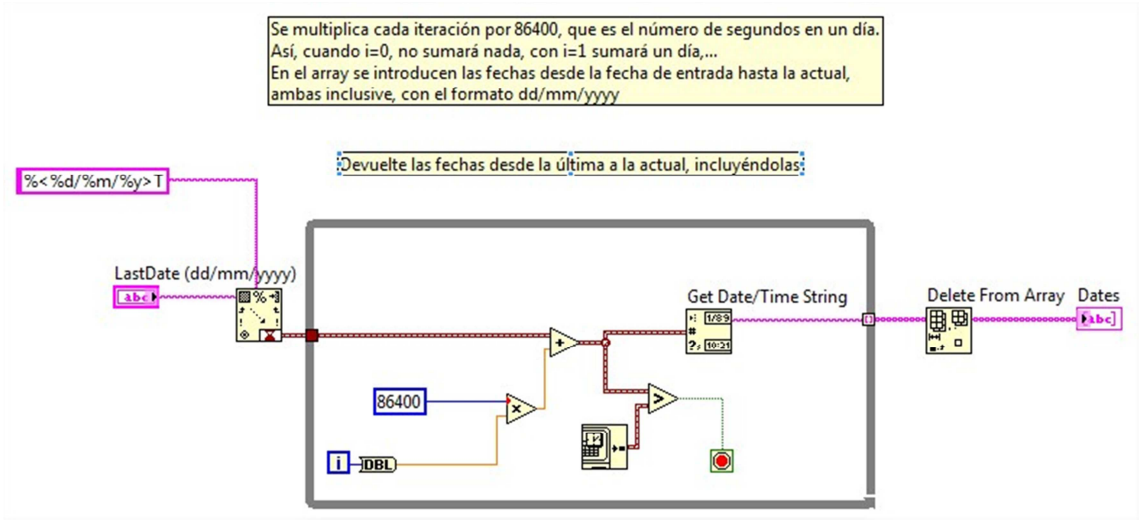

Figura 3.110: diagrama de bloques de ExtractDates.vi

Antes de iniciarse el bucle, la fecha de entrada se convierte a timestamp. En cada iteración, se multiplica el iterador (i) por 86400 (número de segundos en un día) y se suma a la fecha de inicio. Mientras esa fecha no sea estrictamente mayor a la actual, se añadirá al array de salida. En la última iteración, al terminar el bucle, se incluye la fecha que es inmediatamente siguiente a la actual, por lo tanto se elimina.

Podemos ver un ejemplo de funcionamiento en la Figura 3.111.

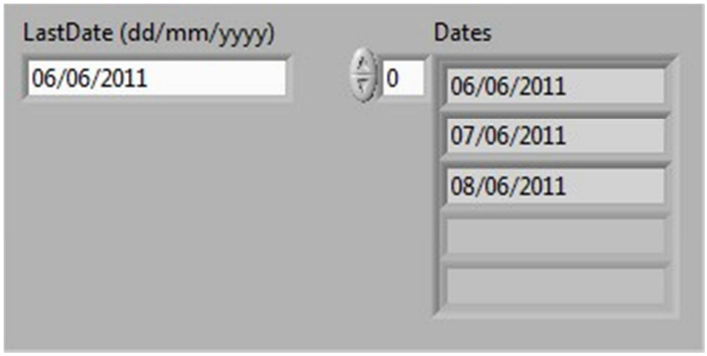

Figura 3.111: Ejemplo de funcionamiento de ExtractDates.vi

ii. Media

El SubVI Media.vi calcula los valores promedios de un array de dobles según la cantidad de datos especificada para hacer la media.

Icono: 

Representa la palabra MEDIA, que es la operación matemática que realiza.

Argumentos de entrada:
-Array: array de números dobles

-Cantidad: representa la cantidad de datos entre los que se hará la media. Si el valor es 5, se hará la media de los números en las posiciones 0-4, 5-9,0-14,…

Argumentos de salida:
-Result: resultado de realizar la operación media al array de entrada

El funcionamiento del SubVI se puede ver en la Figura 3.112.

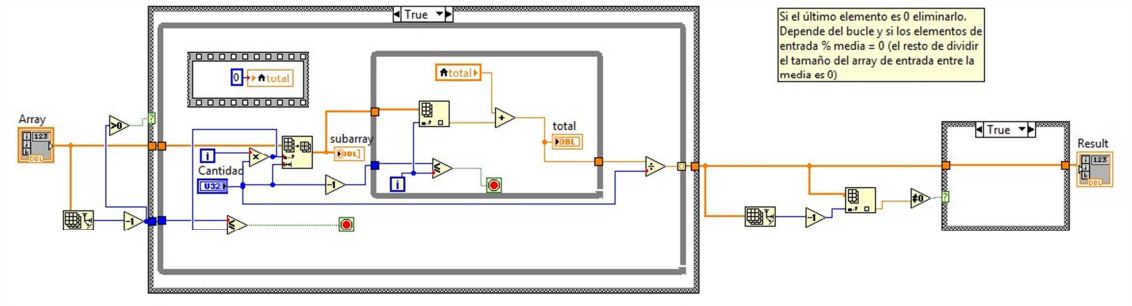

Figura 3.112: diagrama de bloques del subVI Media.vi

El SubVI consta principalmente de dos bucles anidados, que se pueden apreciar en la Figura 3.113.

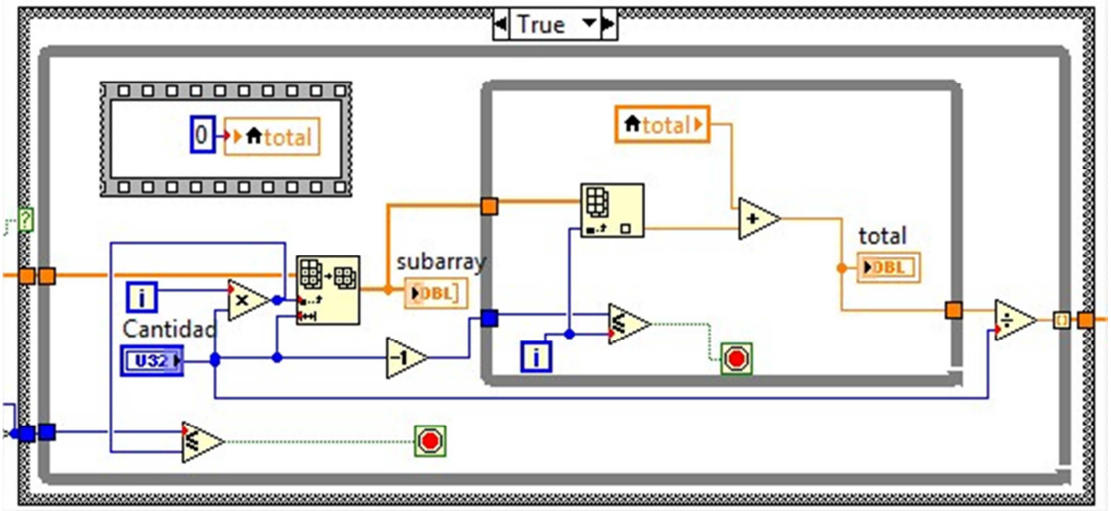

Figura 3.113: Bucles anidados del SubVI Media.vi

En el bucle externo, por cada iteración, se extrae del array de entrada un subarray de tantas posiciones como indique Cantidad. El primer elemento del subarray a recoger estará en la posición i*Cantidad, es decir, siendo i=0, y Cantidad=3, se cogerá el subarray con los elementos que se encuentren en las posiciones 0,1 y 2. Siendo i=1 se recogerán los elementos en las posiciones 3(i*Cantidad),4 y 5.

Ese subarray se trata en el bucle interno, en el que se suman los valores de sus elementos (desde 0 hasta Cantidad-1) y se divide entre el número de ellos (Cantidad).

Por último, Depende del bucle y si los elementos de entrada % media = 0 (el resto de dividir el tamaño del array de entrada entre la media es 0), puede generarse un elemento de más de valor 0. En ese caso, se eliminará.

Podemos ver un ejemplo de funcionamiento en la Figura 3.114.

Figura 3.114: Ejemplo de funcionamiento del subVi Media.vi

iii. SearchLostTime

El SubVI SearchLostTime.vi genera un array con los tiempos que faltan en otro pasado como entrada, hasta que llega al momento actual.

Icono: 

Representa el cálculo de horas llevado a un array.

Argumentos de entrada:
-Now: Timestamp con la hora actual
-Last Hour Array: array de cadenas de caracteres con horas en formato hh:mm:ss

Argumentos de salida:
-Lost Time Array: array de cadena de caracteres con 3600 posiciones, correspondientes a la última hora. Si una hora ya estaba en el array de entrada, aparecerá en blanco. Si no, mostrará la hora correspondiente. Este array está ordenado a la inversa.

El funcionamiento del SubVI se puede ver en la Figura 3.115.

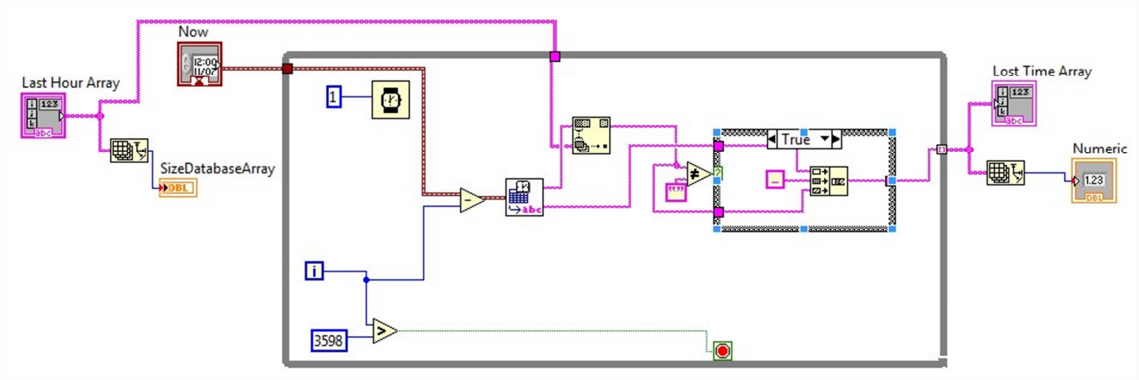

Figura 3.115: diagrama de bloques del subVI SearchLostTime.vi

El SubVI consta de un único bucle. En este bucle, se va restando un segundo a la hora pasada como argumento, hasta retroceder una hora. En cada iteración, se comprueba si el dato existe en el array pasado como entrada. Si existe, se inserta un espacio en blanco. Si no, esa hora se concatena con la fecha actual y se inserta en el array de salida.

Podemos ver un ejemplo de funcionamiento en la Figura 3.116.

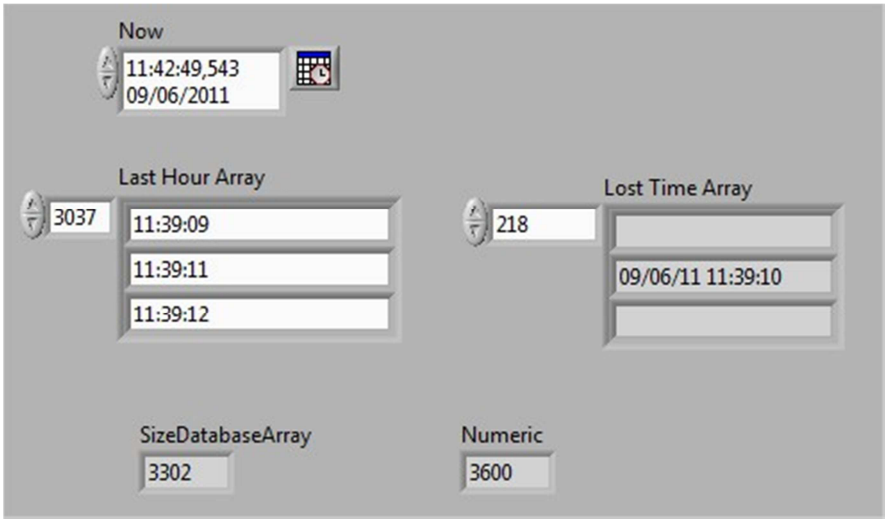

Figura 3.116: Ejemplo de funcionamiento de SearchLostTime

iv. SelectedArray

El SubVI SelectedArray.vi obtiene a partir de un array de cadenas de caracteres y otro array con booleanos, se obtiene un array con el nombre de las cadenas de caracteres cuya posición coincida con un valor TRUE en el array de booleanos.

Icono: 

Representa un array de booleanos (check box) traspasado a otro array.

Argumentos de entrada:
-Boolean Array: array de booleanos
-Variables: Array de cadenas de caracteres

Argumentos de salida:
-SelectedColumns: array de cadenas de caracteres que coinciden con posiciones con valor TRUE en el array de booleanos

El funcionamiento del SubVI se puede ver en la Figura 3.117.

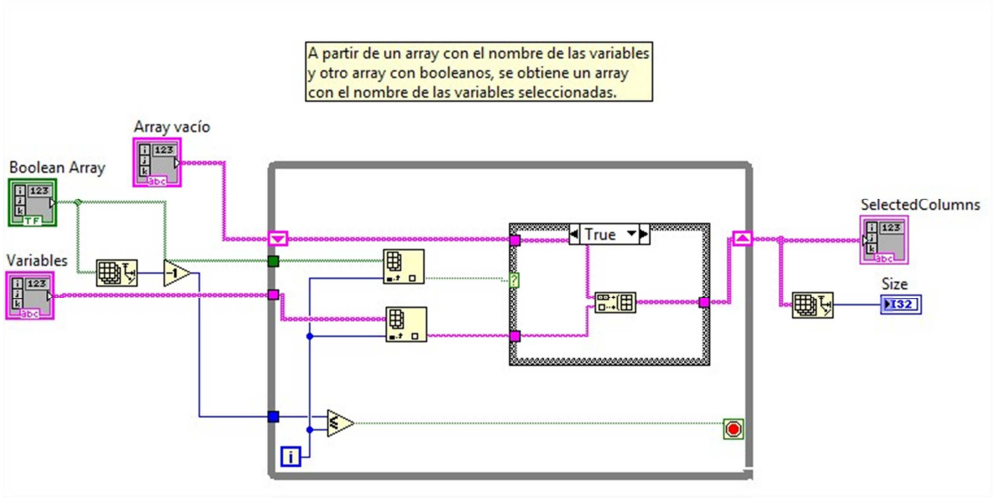

Figura 3.117: diagrama de bloques del SubVI SelectedArray.vi

Consta de un único bucle que se ejecuta para cada posición del array de booleanos. En cada caso, si devuelve true, se inserta en el array de salida el valor correspondiente a esa misma posición del array de cadenas de caracteres.

Podemos ver un ejemplo de funcionamiento en la Figura 3.118.

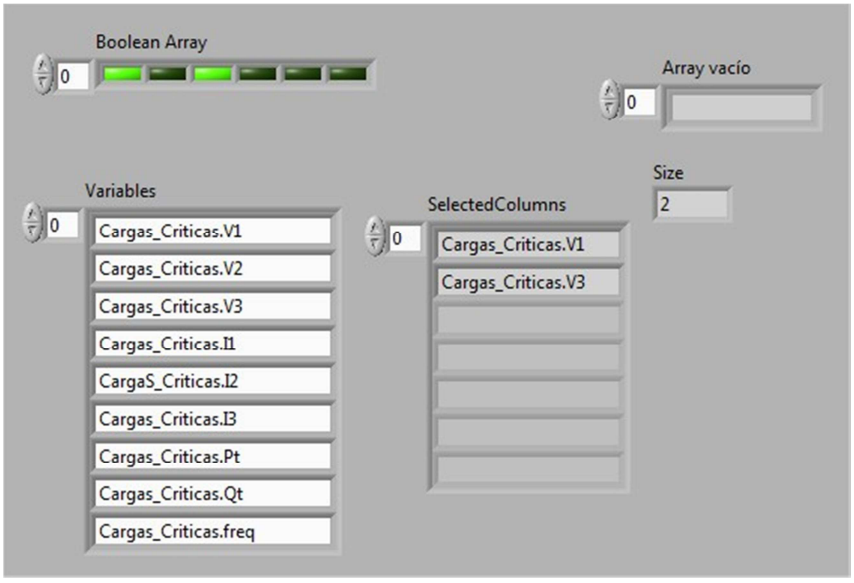

Figura 3.118: Ejemplo de funcionamiento de SelectedArray

v. SelectedTables

Si el array de cadenas de caracteres tiene un tamaño mayor que 0, introduciremos el valor actual en el array de salida. Se usa en los Históricos y SelectedData para calcular las tablas de las que se ha seleccionado algún dato.

Icono: 

Representa una interrogación sobre si el dispositivo actual correspondiente a un módulo de la base de datos, está seleccionado.

Argumentos de entrada:
-In Array: Array de cadenas de caracteres con las tablas ya seleccionadas
-Table: cadena de caracteres con el nombre de tabla actual
-Col Selected: array de cadenas de caracteres con las variables seleccionadas de la tabla actual

Argumentos de salida:
-SelectedTables: array de cadenas de caracteres con las tablas seleccionadas, incluyendo o no la actual

El funcionamiento del SubVI se puede ver en la Figura 3.119.

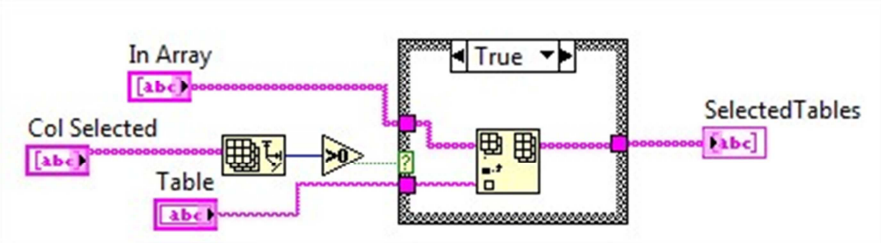

Figura 3.119: diagrama de bloques del SubVI SelectedTables.vi

Comprueba si el tamaño del array de las columnas es mayor que 0, es decir, si por lo menos se ha seleccionado alguna columna. En ese caso, se añadirá la tabla actual al final del array con las tablas ya seleccionadas (In Array). En caso contrario, no se añadirá y la salida corresponderá a In Array.

Podemos ver un ejemplo de funcionamiento en la Figura 3.112.

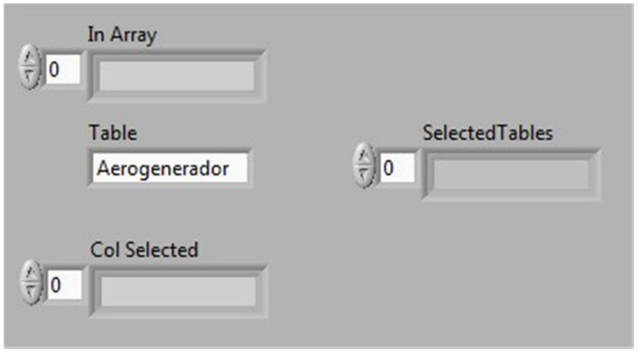

Figura 3.120: Ejemplo de funcionamiento de SelectedTables

vi. TrimArray

El SubVI TrimArray.vi elimina, a partir de un array y de una posición, esa posición si se encuentra en blanco. Devuelve el array con la posición eliminada o no, según sea el caso.

Icono: 

Representa la acción cortar sobre un array

Argumentos de entrada:
-Index: índice a comprobar
-LostTimeArray: Array de cadenas de caracteres, en este caso, de 3600 posiciones, que estarán en blanco si esa fecha y hora ya existe en la base de datos.

Argumentos de salida:
-NextIndex: siguiente índice a comprobar
-Trimmed Lost Time: array de entrada con la posición correspondiente eliminada, en su caso.

El funcionamiento del SubVI se puede ver en la Figura 3.121.

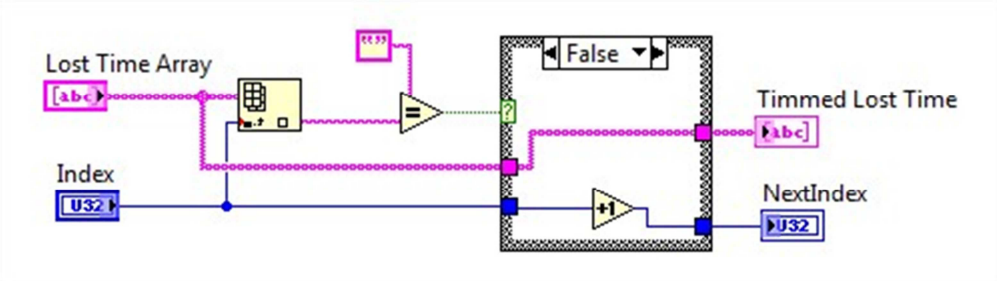

Figura 3.121: Diagrama de bloques del subVI TrimArray

Simplemente comprueba si para la posición dada, el elemento coincide con cadena vacía. Si es así, elimina esa posición y devuelve el mismo índice, para comprobar el elemento siguiente. Si no es vacío, no elimina nada y el elemento siguiente se encontrará en la posición actual +1.

Este SubVI se ejecuta en un bucle, una vez para cada posición del array de entrada. El entorno de ejecución se pude ver en la Figura 3.122.

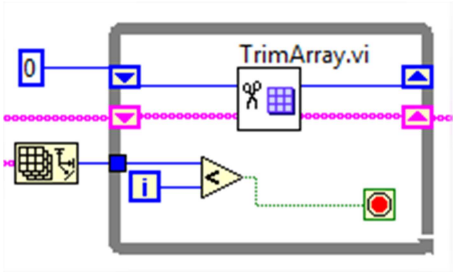

Figura 3.122: Entorno de ejecución de TrimArray.vi

El índice que devuelve se toma como índice de entrada en el bucle que se ejecuta para cada posición.

Podemos ver un ejemplo de funcionamiento en la Figura 3.123. En este caso, el índice a comprobar es el 0, y, como corresponde a una posición vacía, se elimina y el siguiente índice a comprobar será el mismo, ya que ahora corresponde a otro elemento.

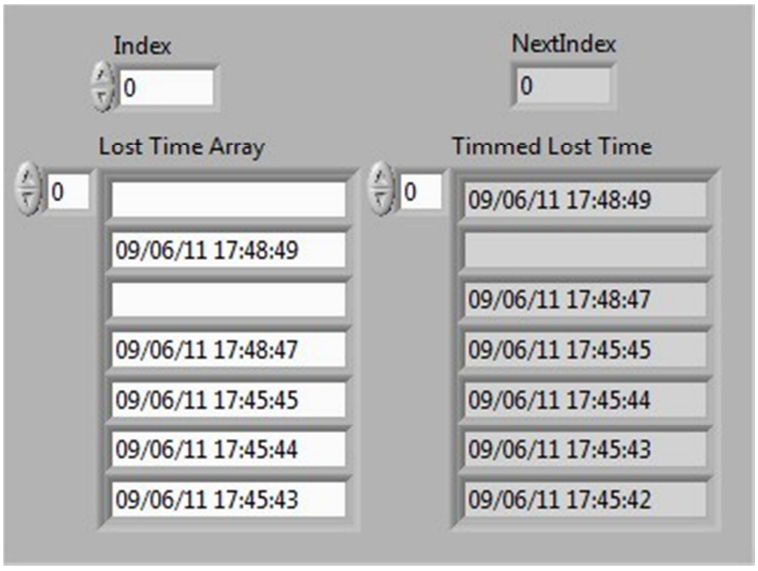

Figura 3.123: Ejemplo de funcionamiento de TrimArray

d) DBToolsLLB

Agrupa las librerías que manejan datos que o bien provienen o bien van a parar a una base de datos. Es decir, son herramientas para el manejo de datos relacionados con una base de datos.

SubVIs pertenecientes a la librería:

i. AddColumn

Este SubVI sirve como interfaz para que el usuario añada una columna a una tabla ya existente en la base de datos Microrred.

Icono: 

Representa la acción nueva columna sobre una base de datos.

No tiene argumentos de entrada ni de salida. Este SubVI es llamado desde MenuDB (3.2.9) como ventana emergente y se cierra automáticamente al terminar su ejecución.

El funcionamiento del SubVI se puede ver en la Figura 3.124.

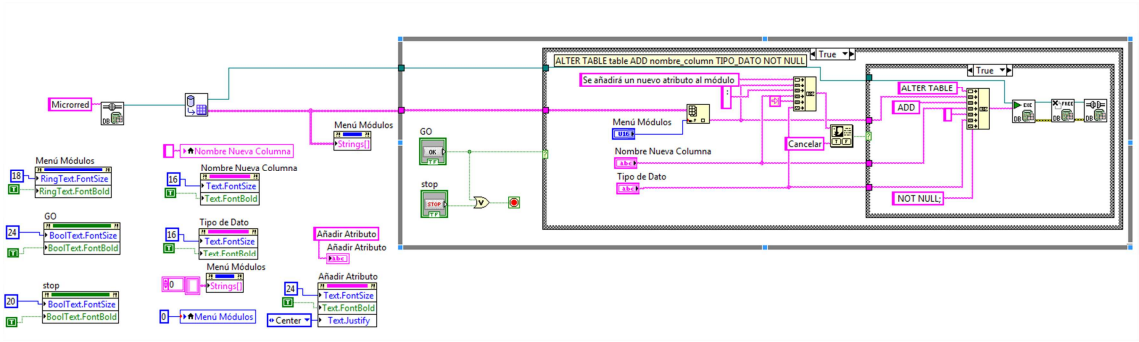

Figura 3.124: Diagrama de bloques de AddColumn

Antes del bucle while se ejecutan unos parámetros de configuración de los menús, indicadores y botones. Cabe resaltar el uso de la función ListTables, ubicada en esta misma librería, para rellenar el menú con la lista de módulos en la base de datos, como se ve en la Figura 3.125.

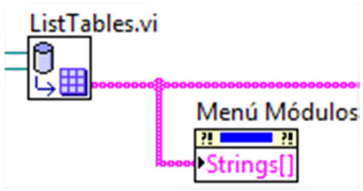

Figura 3.125: uso de ListTables

El bucle terminará una vez hayamos pulsado sobre GO o sobre SALIR. Sin embargo, si pulsamos sobre el primero, se ejecutará la primera de las estructuras case, cuyo funcionamiento se ve en la Figura 3.126.

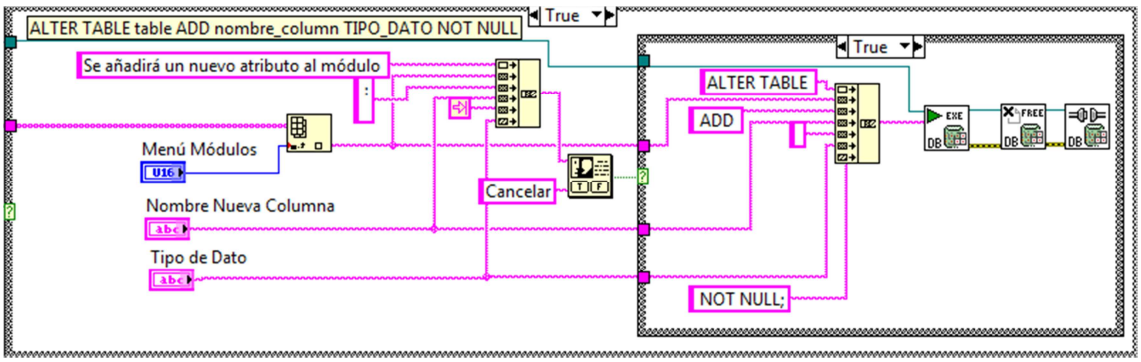

Figura 3.126: estructuras CASE en AddColumn

En este caso, se rescata la información proporcionada por el usuario en los menús e indicadores y se muestra una ventana emergente con la misma. En caso de aceptar, se ejecuta la segunda estructura CASE que enviará a la base de datos una consulta como la siguiente:

ALTER TABLE table1 ADD nombre_column TIPO_DATO NOT NULL;

Con esto conseguiremos que la tabla seleccionada se vea incrementada con una columna, que corresponderá a la especificada por el usuario.

Para ver un ejemplo de funcionamiento dirigirse a la sección 3.2.9.

ii. ArrayToDB

El SubVI ArrayToDB obtiene, a partir de un array, una posición y un tamaño, un clúster preparado para ser insertado en una base de datos.

Icono: 

Representa un array insertándose en una base de datos.

Argumentos de entrada:
-Date: cadena de caracteres con la fecha actual
-Time: cadena de caracteres con la hora actual
-Array: array de números dobles con los datos
-Index: índice a partir del cual nos interesan los datos
-Nº Vars: número de variables de interés

Argumentos de salida
-DataN: Clúster con la fecha, la hora, y luego N datos (valiendo N desde 2 hasta 15)
-Ok: booleano que indica si el dato de error es correcto. En caso de error FALSE.

El funcionamiento del SubVI se puede ver en la Figura 3.127.

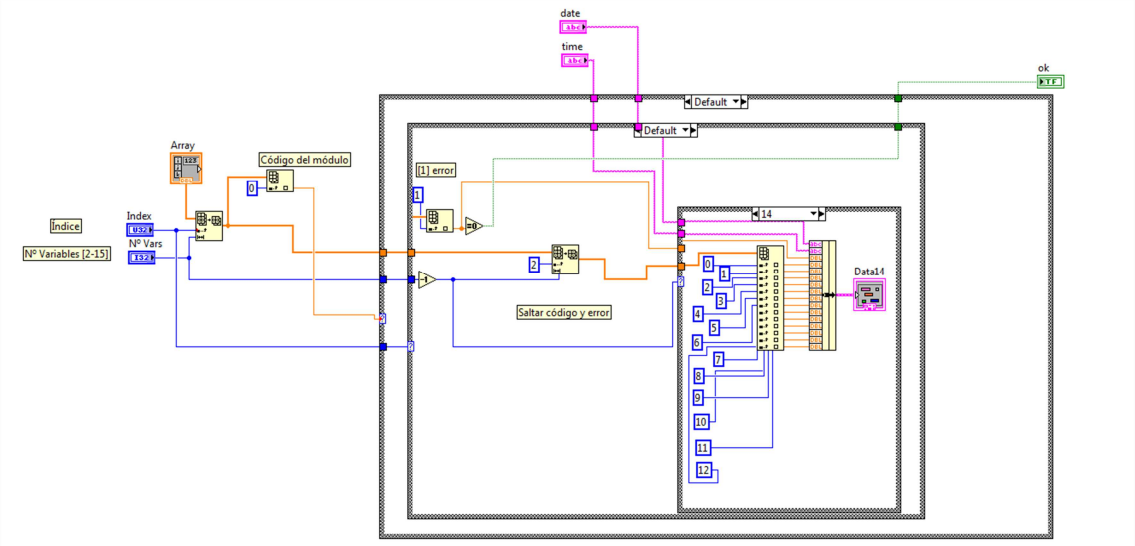

Figura 3.127: diagrama de bloques de ArrayToDB

La primera de las estructuras CASE comprueba si se trata del módulo de variables Meteorológicas, según el valor del código correspondiente en los datos (índice). En caso de ser 9900 se trata de las variables Meteorológicas, entonces, comprobará si cada uno de los datos es distinto de -100 (si hay error) y en caso de que alguno sea -100, la variable ok valdrá FALSE. A los datos se les añade la hora y la fecha.

En los demás casos, se diferencia si se tratan del Híbrido Inversor, Híbrido Eólica o los demás. Por defecto, se salta el código de módulo y el error y, según el número de

variables restantes, la salida será por un DataN en concreto, correspondo N a la cantidad de variables -1. Es decir, si en Nº vars introducimos 11, la salida será por Data10.

En el caso del Inversor Híbrido y el Híbrido Eólica, cogen como dato de error el correspondiente al Híbrido Cargador. Además, tampoco tienen variable que indique el código del módulo, por lo que no tendrán que saltarse ninguna posición, únicamente construir el array añadiendo la fecha y hora y devolverlo.

iii. CompleteArray

El SubVI CompleteArray.vi completa un array de forma que se pueda hacer la media correctamente. Por ejemplo, si el array tiene cinco posiciones y queremos hacer la media cada dos elementos, añadirá un sexto elemento con el mismo valor del 5º, de modo que la media no quede falseada. Es decir, completará un array de modo que el resto de dividir el tamaño entre el número de elementos de la media sea 0.

Icono: 

Representa una medición sobre un array

Argumentos de entrada:
-InputArray: array de números dobles de entrada
-Media: cantidad de valores entre los que se hace la media

Argumentos de salida:
-CompletedArray: array de números dobles con las posiciones completadas

El funcionamiento del SubVI se puede ver en la Figura 3.128.

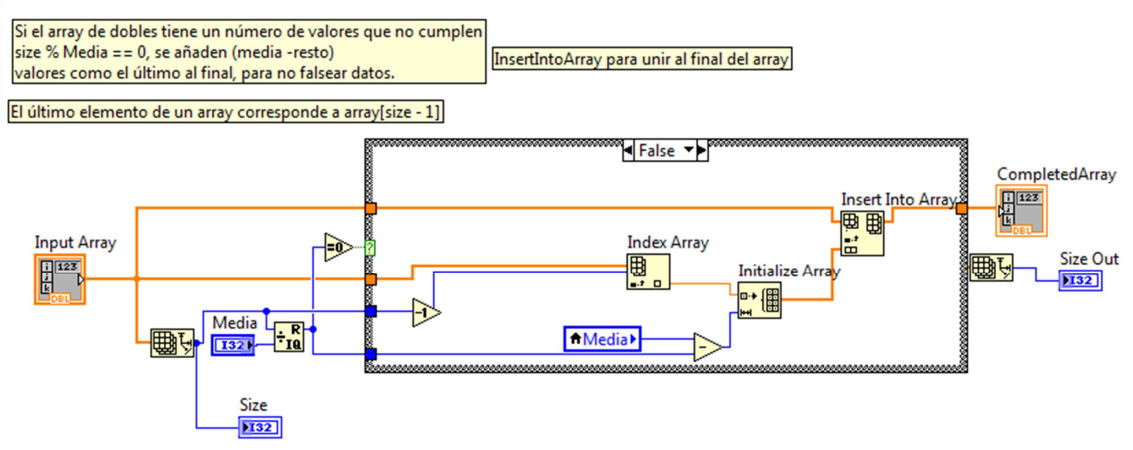

Figura 3.128: diagrama de bloques de CompleteArray

Si el array de dobles tiene un número de valores que no cumplen (size % Media = = 0), se añaden (media - resto) valores como el último al final, para no falsear datos. es decir,

se extraerá un subarray de ese tamaño, inicializado al último valor y se añadirá al final del array de inicio.

Podemos ver un ejemplo de funcionamiento en la Figura 3.129.

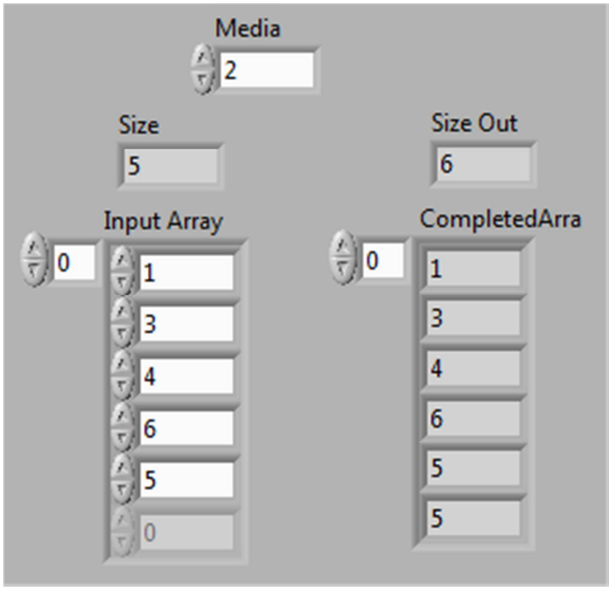

Figura 3.129: ejemplo de funcionamiento de CompleteArray

iv. CreateTable

Este SubVI sirve como interfaz para que el usuario añada una nueva tabla a la base de datos Microrred.

Icono: 

Representa la acción nueva tabla sobre una base de datos.

No tiene argumentos de entrada ni de salida. Este SubVI es llamado desde MenuDB (3.2.9) como ventana emergente y se cierra automáticamente al terminar su ejecución.

El funcionamiento del SubVI se puede ver en la Figura 3.130.

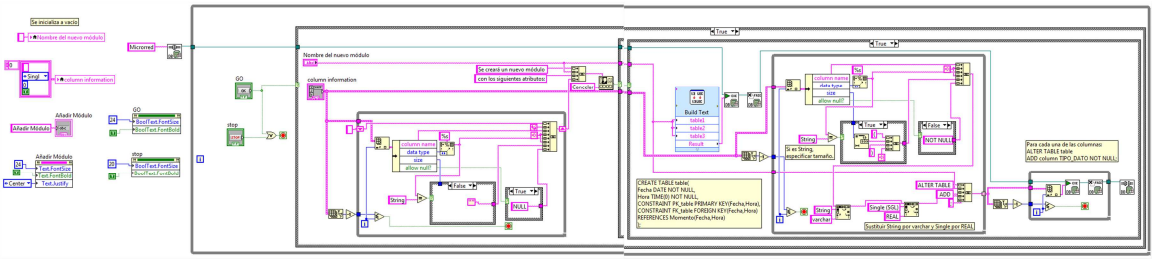

Figura 3.130: diagrama de bloques de CreateTable.vi

Antes del bucle while se ejecutan unos parámetros de configuración de los menús, indicadores y botones.

El bucle terminará una vez hayamos pulsado sobre GO o sobre SALIR. Sin embargo, si pulsamos sobre el primero, se ejecutará la primera de las estructuras case.

En esa primera estructura hay un bucle while que generará, a partir de los datos introducidos por el usuario, un mensaje con la información insertada que aparecerá en una ventana emergente. En caso de aceptar la ventana emergente, se generará una tabla con el nombre especificado usando la siguiente consulta SQL:

```
CREATE TABLE tabla (
Fecha DATE NOT NULL,
Hora TIME(0) NOT NULL,
CONSTRAINT PK_table PRIMARY KEY(Fecha,Hora),
CONSTRAINT FK_table FOREIGN KEY(Fecha,Hora) REFERENCES
Momento(Fecha,Hora)
);
```

Esta consulta crea una nueva relación en la base de datos con el nombre especificado, en este caso tabla, que tiene como atributos la fecha y la hora, que se relacionan, al igual que el resto de relaciones de dispositivos, con la relación Momento. Estos dos atributos serán al mismo tiempo clave primaria de la relación y clave foránea, referenciando a las columnas del mismo nombre de la relación Momento.

Después, para cada uno de los atributos especificados, se ejecuta una consulta como la siguiente:

```
ALTER TABLE tabla ADD column TIPO_DATO NOT NULL;
```

Es decir, vamos añadiendo uno a uno los atributos a la nueva relación creada.

Para ver un ejemplo de funcionamiento dirigirse a la sección 3.2.9.

v. DBToArray

Convierte un array de clústeres con fecha y hora extraído de la base de datos a un array de cadenas de caracteres con la hora.

Icono: 

Representa la conversión entre base de datos a array.

Argumentos de entrada:
-Last Hour Times: array de clústeres con fechas y horas extraídas de la base de datos

Argumentos de salida:
-Last Hour Array: array de cadenas de caracteres con las horas extraídas de la base de datos.

El funcionamiento del SubVI se puede ver en la Figura 3.131.

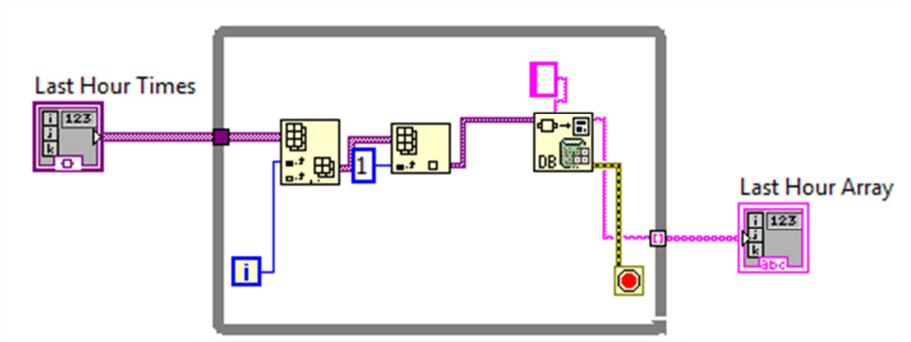

Figura 3.131: diagrama de bloques de DBToArray

Únicamente, para cada posición, convierte el dato a tipo cadena de caracteres. El bucle finaliza cuando la conversión da error.

Podemos ver un ejemplo de funcionamiento en la Figura 3.132.

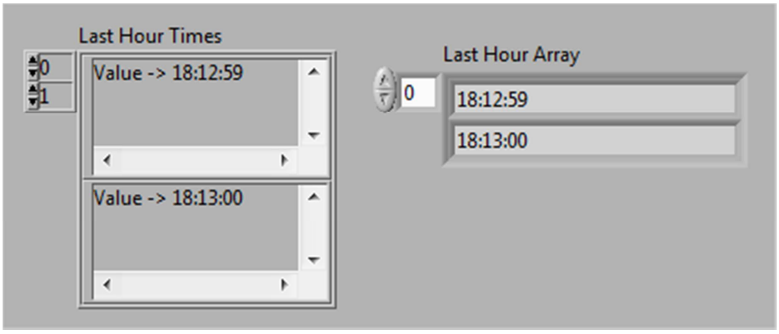

Figura 3.132: ejemplo de funcionamiento de DBToArray

vi. DropColumn

Este SubVI sirve como interfaz para que el usuario elimine una columna de una tabla ya existente en la base de datos Microrred.

Icono: 

Representa la acción eliminar columna sobre una base de datos.

No tiene argumentos de entrada ni de salida. Este SubVI es llamado desde MenuDB (3.2.9) como ventana emergente y se cierra automáticamente al terminar su ejecución. El funcionamiento del SubVI se puede ver en la Figura 3.133.

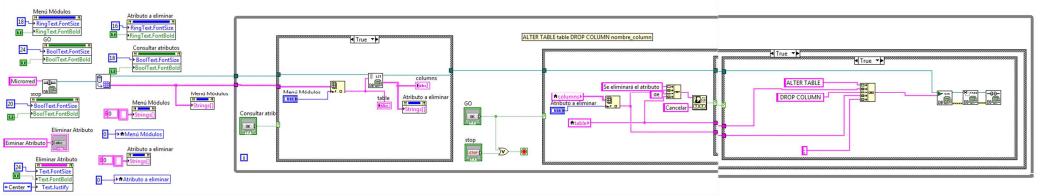

Figura 3.133: diagrama de bloques de DropColumn

Antes del bucle while se ejecutan unos parámetros de configuración de los menús, indicadores y botones. Al entrar al bucle while, una vez se ha pulsado en Consultar Atributos, se ejecuta una estructura case en la que, a partir de la opción elegida en el menú, mostrará sus columnas en el menú correspondiente. Elimina del array de columnas a mostrar las columnas de Fecha y Hora, ya que no se pueden eliminar por formar parte de la clave primaria de la relación.

Una vez se ha elegido la columna a eliminar, si pulsamos en el botón GO, aparecerá una ventana emergente para confirmar la operación, cuyo mensaje se ha obtenido a partir de la información suministrada por el usuario.

Si se acepta esa ventana emergente se ejecuta una consulta SQL de este estilo:

ALTER TABLE tabla DROP COLUMN columna;

Con la que se elimina la columna dada de la tabla especificada.

Para ver un ejemplo de funcionamiento dirigirse a la sección 3.2.9.

vii. DropTable

Este SubVI sirve como interfaz para que el usuario elimina una tabla existente en la base de datos Microrred.

Icono: 

Representa la acción eliminar una tabla sobre una base de datos.

No tiene argumentos de entrada ni de salida. Este SubVI es llamado desde MenuDB (3.2.9) como ventana emergente y se cierra automáticamente al terminar su ejecución. El funcionamiento del SubVI se puede ver en la Figura 3.134.

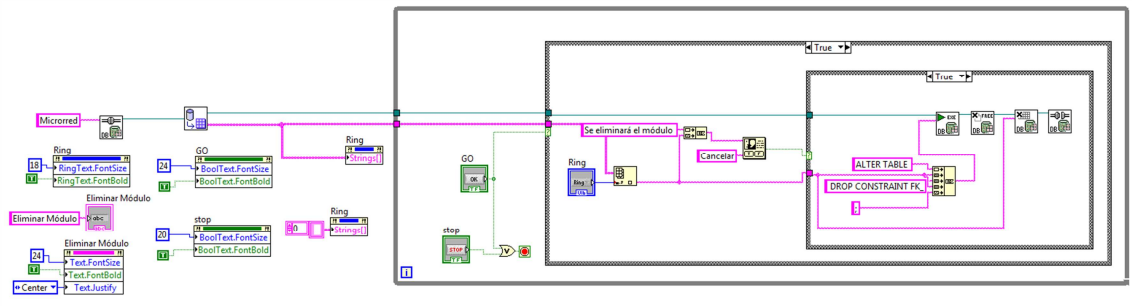

Figura 3.134: diagrama de bloques de DropTable

Antes del bucle while se ejecutan unos parámetros de configuración de los menús, indicadores y botones. Al pulsar el botón GO, se ejecutará una estructura CASE en la cual aparecerá una ventana emergente que indicará el módulo a eliminar. En caso de aceptar

esta ventana emergente, se ejecutará una segunda estructura que generará una consulta SQL de este estilo:

ALTER TABLE tabla DROP_CONSTRAINT FK_table;

Eso eliminará la clave foránea, por lo que la tabla ya no estará ligada al resto de la base de datos. A continuación se elimina la tabla en la función propia de LabVIEW que recoge el nombre de la misma como argumento.

Para ver un ejemplo de funcionamiento dirigirse a la sección 3.2.9.

viii. Get Database Information

Este VI se ha elaborado a partir del ejemplo que ofrece LabVIEW con el mismo nombre, realizando muy pocas modificaciones. Aporta la información general de la base de datos Microrred.

Icono: 

Representa la extracción de información de la base de datos hacia una tabla.

No tiene argumentos de entrada ni de salida. Este SubVI es llamado desde MenuDB (3.2.9) como ventana emergente y se cierra automáticamente al terminar su ejecución. El funcionamiento del SubVI se puede ver en la Figura 3.135.

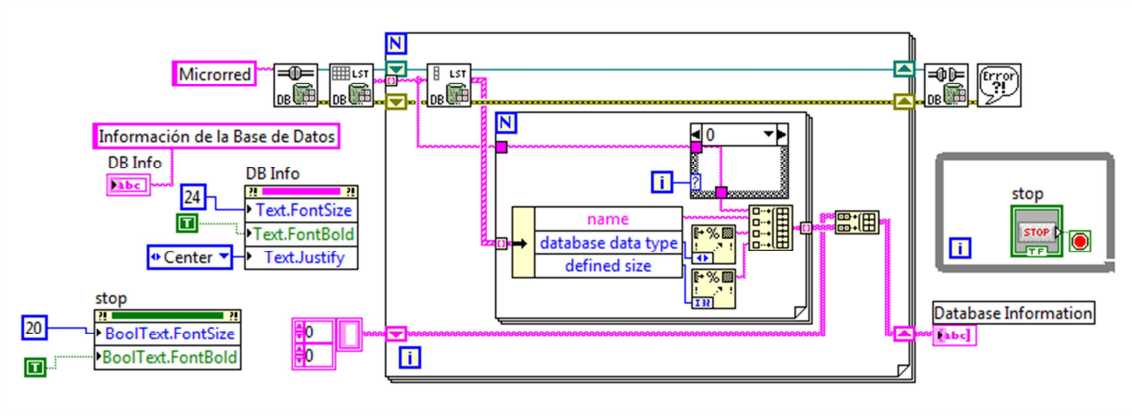

Figura 3.135: diagrama de bloques de Get Database Information

Mediante las funciones de LabVIEW, se extrae la información de las tablas y columnas de la base de datos Microrred y con ello se rellena una tabla.

Existe un botón dentro de un bucle infinito para que la información se muestre hasta que pulsemos el botón para salir.

Para ver un ejemplo de funcionamiento dirigirse a la sección 3.2.9.

ix. InsertDB

Este SubVI inserta los datos en la base de datos a partir de un array de números dobles.

Icono: 

Representan los números dobles siendo insertados en la base de datos.

Argumentos de entrada:
-Array: array de números dobles a insertar en la base de datos

No tiene argumentos de salida.

Este SubVI se ha creado a partir del Esquema de Comunicación y corresponde a toda la parte de ejecución del mismo, que se puede ver en el apartado 3.2.4.

Inserta el array de dobles pasados como argumento de entrada. Además, en caso de que ya existan los datos en la base de datos, no saltará una excepción de error, si no que será ignorada, como se puede ver en la Figura 3.136.

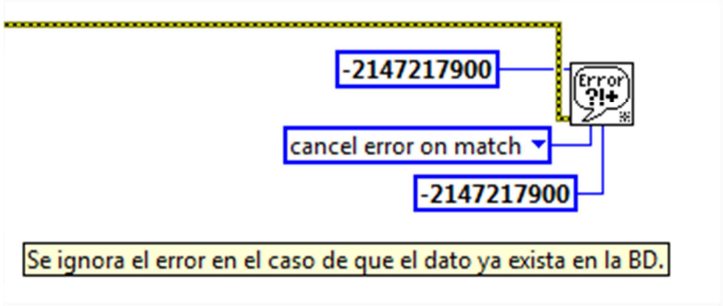

Figura 3.136: Se ignora el error que indica que el dato ya existe en la Base de Datos.

x. InsertTDMS

A partir de los datos de un fichero TDMS proveniente del Pxi, inserta los datos en la base de datos Microrred.

Icono: 

Representa la acción de insertar los datos del TDMS en la base de datos.

Argumentos de entrada:
-OutputArray: Matriz de números dobles con los datos a insertar
-LostTimeArray(Opcional): array de cadenas de caracteres con las fechas y horas a insertar

No tiene argumentos de salida.

El funcionamiento del SubVI se puede ver en la Figura 3.137.

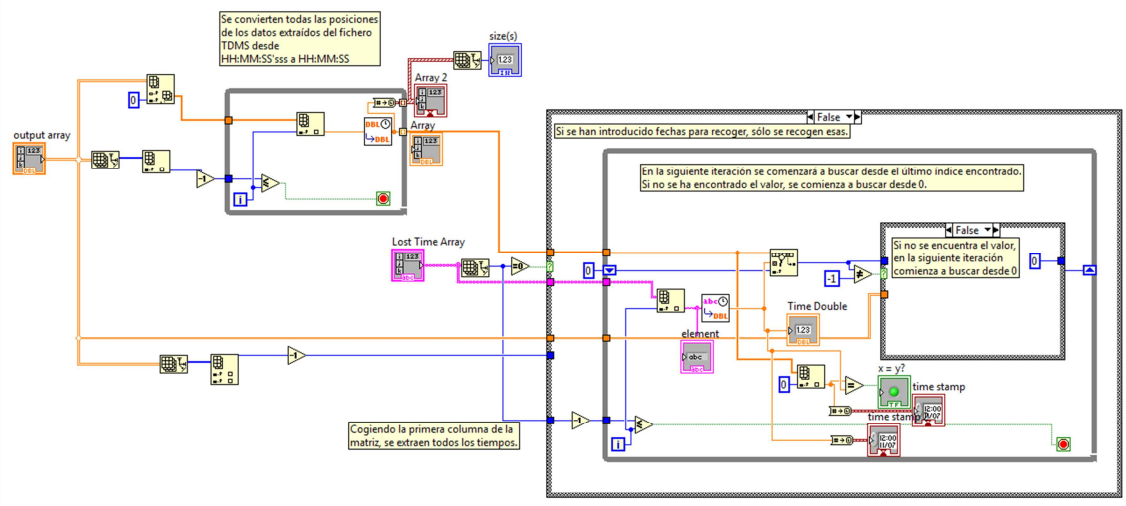

Figura 3.137: diagrama de bloques de InsertTDMS

Primero se ejecuta un bucle con los datos pertenecientes a las horas (columna 0 de la matriz) y, mediante el uso de la función creada DoubleWithoutMilliseconds se convierten todas las posiciones de los datos extraídos del fichero TDMS desde HH:MM:SS'sss a HH:MM:SS.

A continuación se ejecuta una estructura CASE. Si hemos introducido el argumento opcional con las fechas a encontrar, buscará únicamente esas. Si no, insertará todas las posiciones.

En caso de insertar solo las especificadas, éstas se convierten mediante la función creada TimeStringToDouble a número doble, y ese número se busca dentro del array de horas que hemos convertido. Si existe, devuelve el índice, se inserta, y se comienza a buscar a partir del índice siguiente, ya que las horas están ordenadas cronológicamente.

Para insertar los datos se usa la función creada InsertDB.vi.

xi. ListTables

Este SubVI lista devuelve una array de cadenas de caracteres con las tablas de la base de datos especificada en la conexión.

Icono: 

Representa la acción de volcar la información de la base de datos en un array.

Argumentos de entrada:
-Connection reference: conexión ya establecida con una base de datos

Argumentos de salida:
-Connection reference out: conexión de salida con una base de datos

-Tables: array de cadenas de caracteres con el nombre de las tablas existentes en la base de datos

El funcionamiento del SubVI se puede ver en la Figura 3.138.

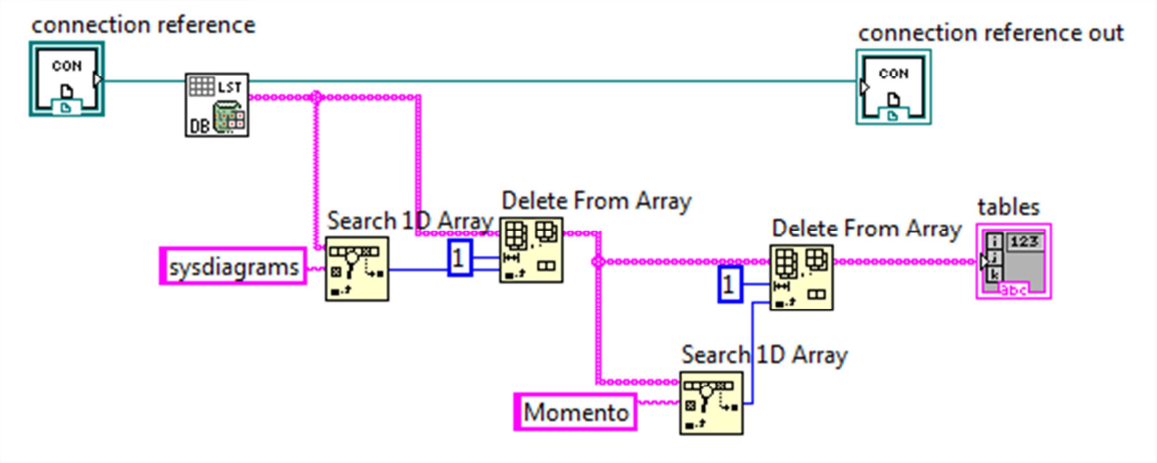

Figura 3.138: diagrama de bloques de ListTables

Se usa la función de LabVIEW que lista las tablas y, al resultado, se le eliminan las tablas correspondientes a sysdiagrams y Momento. Este SubVI será usado para mostrar las tablas al usuario (en MenuDB, apartado 3.2.9), y no interesa que tenga acceso a tablas a las que solo debería acceder el administrador, como sysdiagrams. Tampoco interesa que realice cambios en la tabla Momento, ya que forma una parte importante de la estructura de la base de datos. Así, el usuario solo accede a los nombres de las tablas correspondientes a dispositivos de la Microrred, y serán esas las que pueda modificar o eliminar.

xii. *SelectLastHour*

Este SubVI selecciona los datos de las horas base de datos correspondiente a la última hora.

Icono: 

Representa un reloj con un -1, para indicar una hora anterior, y una base de datos.

No tiene argumentos de entrada.

Argumentos de salida:
-Now: momento actual, al cual se le ha restado 1 hora extraer los datos de la base de datos.
-LastHourTimes: horas de las que existen datos, en los últimos 60 minutos

El funcionamiento del SubVI se puede ver en la Figura 3.139.

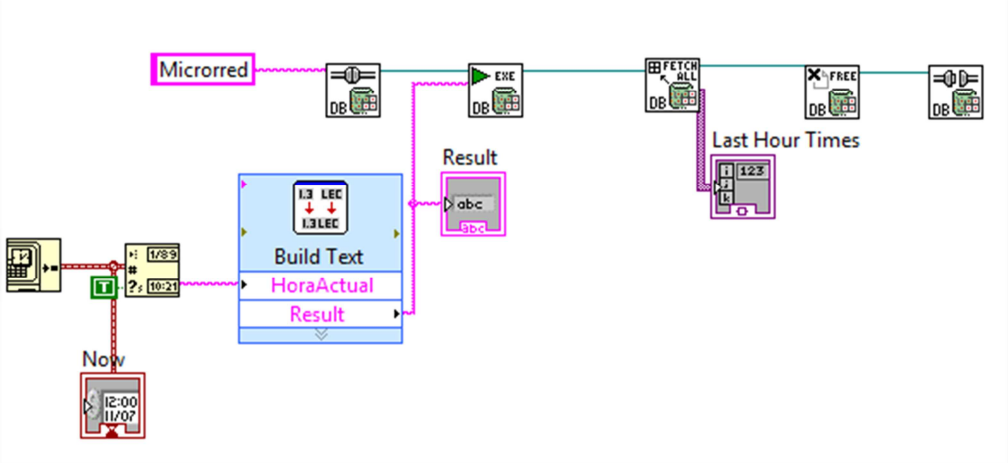

Figura 3.139: diagrama de bloques de SelectLastHour

Se ejecuta una consulta SQL de este tipo:

```
SELECT  CONVERT(varchar,Hora,8) Hora
FROM Momento
WHERE Fecha = convert(date,getdate())
AND Hora > convert (varchar, dateadd(hour,-1,'hora_actual'),8)
ORDER BY Fecha, Hora
```

Esta consulta selecciona las horas, convertidas a cadena de caracteres, de la tabla Momento, que contiene las Fechas y Horas en las cuales se han producido datos. Así, selecciona las horas del día actual (getdate()) y cuyas horas sean mayores que la actual restándole 1.

xiii. SqlServerDateFormat

Este SubVI convierte el formato de fechas utilizado por LabVIEW (DD/MM/YYYY) al utilizado por el SGBD SQLServer (YYYY-MM-DD).

Icono: 

Representa la conversión de la fecha.

Argumentos de entrada:
-Date DD/MM/YYYY: cadena de caracteres con una fecha expresada en formato DD/MM/YYYY

Argumentos de salida:
-Date YYYY-MM-DD: cadena de caracteres con la misma fecha que la introducida como argumento de entrada, pero expresada en formato YYYY-MM-DD.

El funcionamiento del SubVI se puede ver en la Figura 3.140.

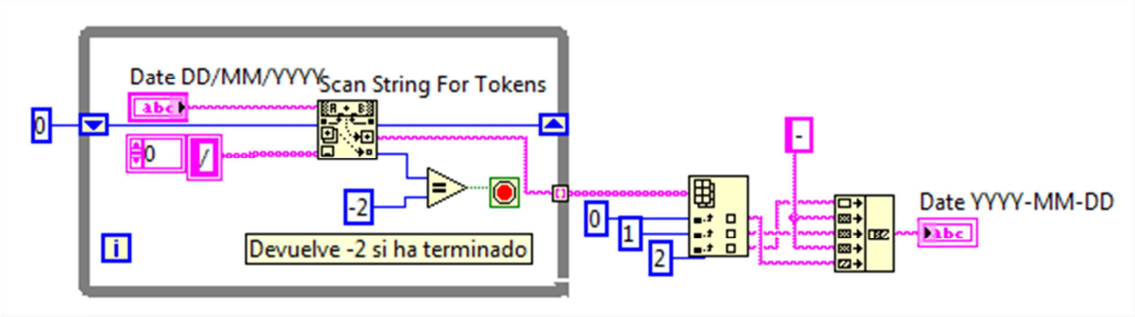

Figura 3.140: diagrama de bloques de SqlServerDateFormat

El SubVI hace uso de la función de LabVIEW ScanStringForTokens. A esta función se le pasa como argumento el string a separar y los separadores, en este caso, el slash(/). Los tokens (pedazos) los va almacenando en un array a la salida del bucle. Cuando la función devuelve -2, significa que ha terminado. Como se supone que la fecha de entrada sigue el formato especificado, sabemos que obtenemos 3 tokens, el primero con el día, el segundo con el mes y el último con el año. Simplemente los concatenamos en el orden correcto con el carácter guión (-) y devolvemos el resultado.

Podemos ver un ejemplo de funcionamiento en la Figura 3.141.

Figura 3.141: ejemplo de funcionamiento de SqlServerDateFormat

xiv. TDMStoDatabase

Este SubVI se basa en el ejemplo proporcionado por LabVIEW llamado ReadTDMS. A partir de un fichero de tipo TDMS obtiene la matriz de datos que están contenidos en él.

Icono: 

Representa la acción de extraer los datos desde el fichero TDMS para que se inserten en una base de datos.

Argumentos de entrada:
-File path: path con el fichero TDMS a leer
-LostTimeArray(Opcional): array con las horas que deseamos recuperar

No tiene argumentos de salida.

El funcionamiento del SubVI se puede ver en la Figura 3.142.

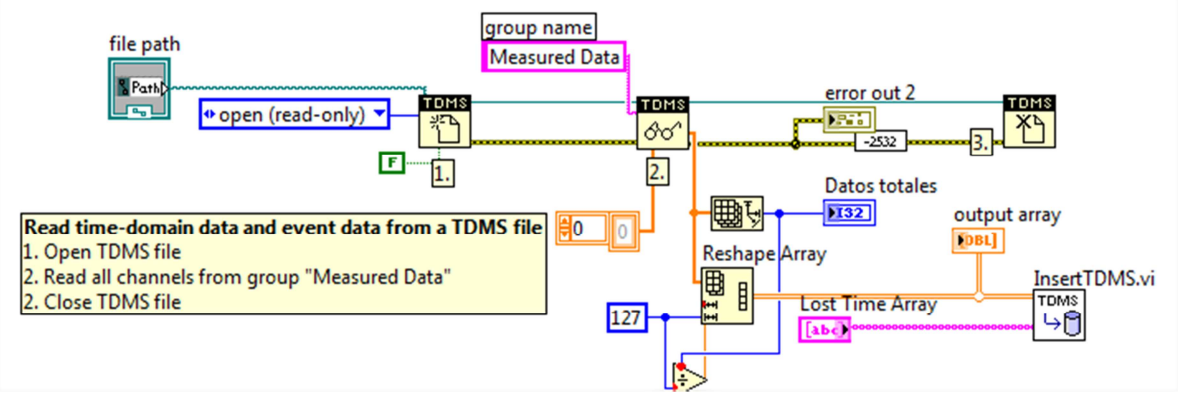

Figura 3.142: diagrama de bloques de TDMStoDatabase

El código abre en lectura el fichero TDMS y lee el grupo de datos que corresponden a los medidos. Después, los datos son recuperados como un único array, pero interesa tenerlo como matriz de datos, en el que cada fila corresponda a los datos de un único momento.

Así, usamos la función ReshapeArray asignando el valor ya conocido de número de datos para que convierta el array en matriz, y esos datos serán mandados a la función creada InsertTDMS, para que sean insertados en la base de datos.

xv. xToArray

Este SubVI convierte un clúster de datos provenientes de una base de datos a un array de números dobles, extrayendo también la fecha y la hora en formato de cadena de caracteres.

Icono: 

Representa la conversión de un clúster de datos, sea cual sea su tamaño, a un array que será representado en una gráfica.

Argumentos de entrada:
-Recordset Data: clúster de datos que tiene en la primera posición la fecha, en la segunda la hora y en las demás valores dobles correspondientes a distintos datos.
-Index: índice del cuál hay que escoger la fila adecuada sobre los datos de la base de datos

Argumentos de salida:
-Date: cadena de caracteres con la fecha.
-Time: cadena de caracteres con la hora.
-Array: array de números dobles con los datos.

El funcionamiento del SubVI se puede ver en la Figura 3.143.

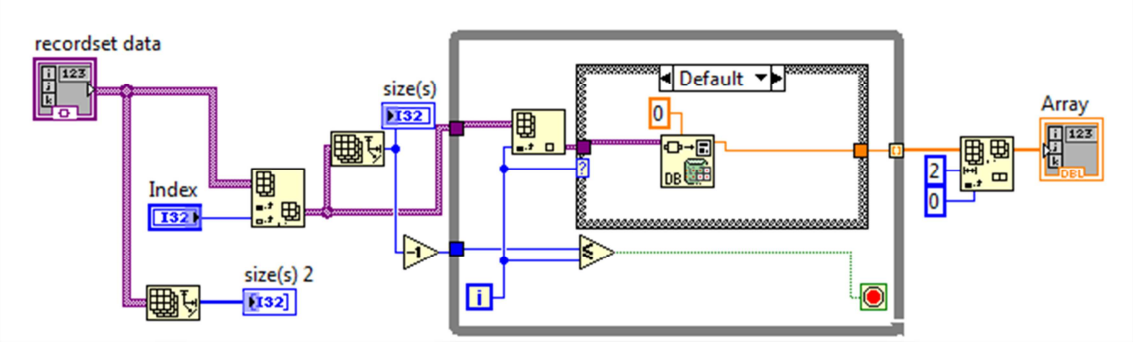

Figura 3.143: diagrama de bloques de xToArray

Antes de entrar al bucle, se extrae la fila de interés de los datos de la base de datos. Después, para cada una de las posiciones del array, se ejecuta una iteración:

-En caso de la primera posición (i = = 0), el elemento corresponderá a la fecha, por lo que se extrae su valor y se almacena en la variable de salida, como podemos ver en la Figura 3.144.

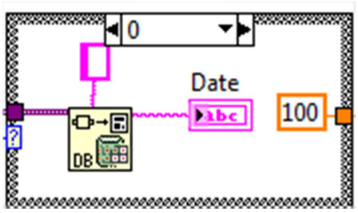

Figura 3.144: Extracción de la fecha

-En caso de la segunda posición (i = = 1), el elemento corresponderá a la hora, por lo que se extrae su valor y se almacena en la variable de salida, como podemos ver en la Figura 3.145.

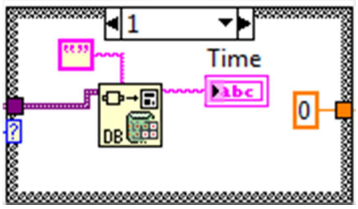

Figura 3.145: Extracción de la hora

-Para el resto de valores, el dato se convierte a tipo doble.

Al finalizar, hemos rellenado todas las posiciones del array, pero también las dos primeras, que correspondían a la fecha y la hora. Por tanto, eliminamos esas dos posiciones y devolvemos el array con únicamente los datos.

El funcionamiento del SubVI se puede ver en la Figura 3.146.

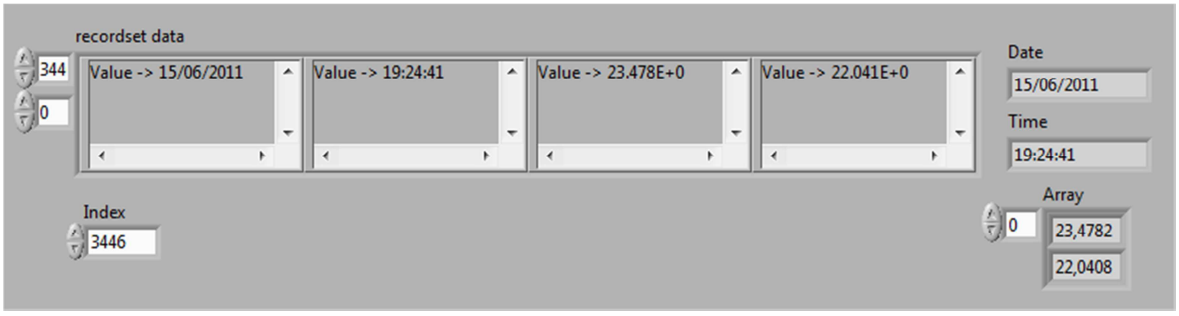

Figura 3.146: diagrama de bloques de xToArray.

xvi. xToString

Este SubVI convierte un clúster de datos provenientes de una base de datos a una cadena de caracteres con los elementos separados por tabuladores.

Icono: 

Representa la conversión de un clúster de datos, sea cual sea su tamaño, a una cadena de caracteres.

Argumentos de entrada:
-Recordset Data: clúster de datos que tiene en la primera posición la fecha, en la segunda la hora y en las demás valores dobles correspondientes a distintos datos.
-Index: índice del cuál hay que escoger la fila adecuada sobre los datos de la base de datos

Argumentos de salida:
-String: cadena de caracteres con los elementos de esa posición separados por tabuladores.

El funcionamiento del SubVI se puede ver en la Figura 3.147.

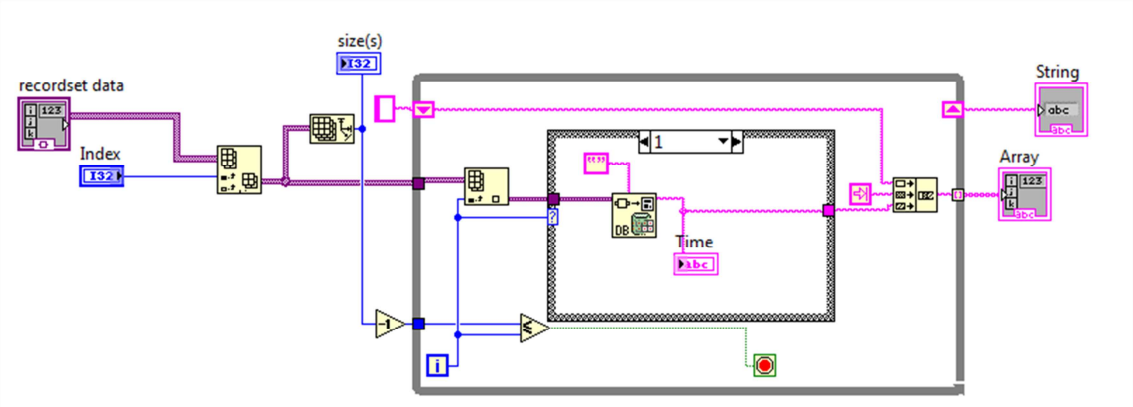

Figura 3.147: diagrama de bloques de xToString

Cada uno de los valores se convierte a cadena de caracteres y se unen con la cadena obtenida hasta ahora (inicializada a cadena vacía) y un carácter tabulador.

El funcionamiento del SubVI se puede ver en la Figura 3.148.

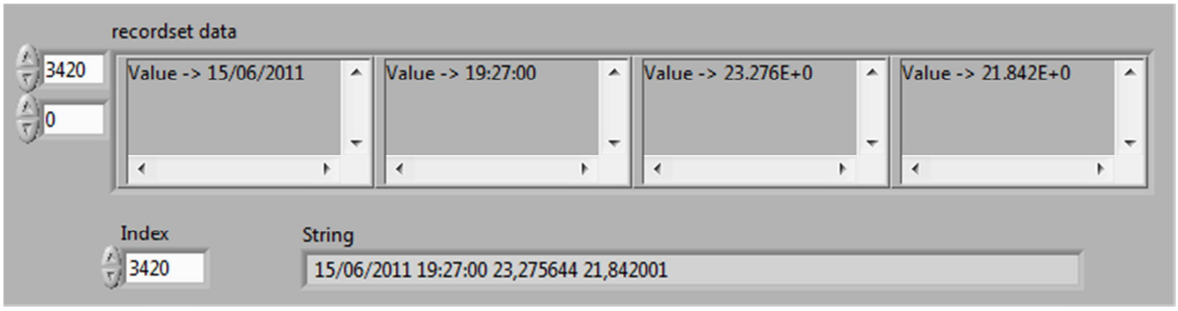

Figura 3.148: diagrama de bloques de xToString

e) FileToolsLLB

Agrupa las librerías que se relacionan con ficheros para realizar sus acciones.

i. *CreateFilename*

Crea un nombre de fichero único y con la extensión especificada. El formato del nombre del fichero dependerá de la hora actual del sistema, siguiendo el modelo: DD_MM_YYYY_hh_mm_ss.extensión

Icono: 

Representa la acción de crear el nombre de un nuevo fichero.

Argumentos de entrada:
-Extensión: cadena de caracteres con la extensión deseada del fichero.

Argumentos de salida:
-Appended path: path completo del futuro nuevo fichero.

El funcionamiento del SubVI se puede ver en la Figura 3.149.

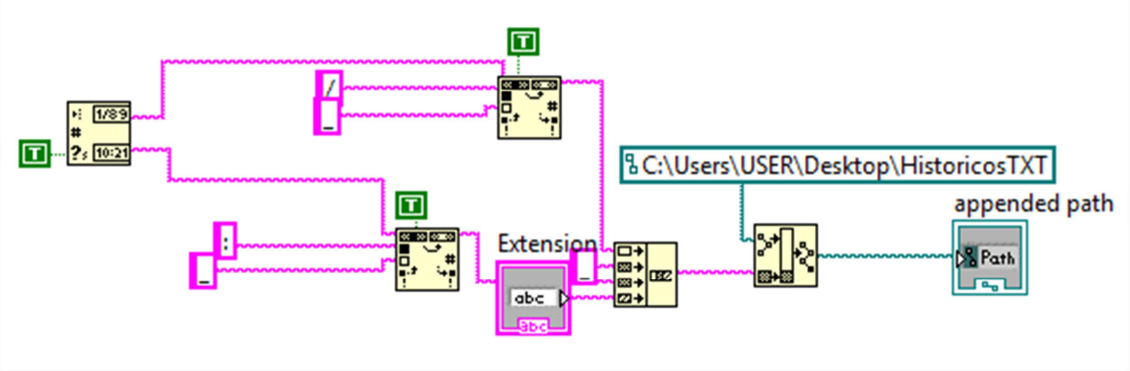

Figura 3.149: diagrama de bloques de CreateFilename

El nombre del fichero se calcula a partir de la hora actual. Se extrae por un lado la fecha y por otro la hora, y se sustituyen los separadores por la barra baja (_). Se concatena con la

extensión introducida por el usuario y eso se añade al path por defecto en el que serán alojados los ficheros (C:\Users\USER\Desktop\HistoricosTXT).

Podemos ver un ejemplo de funcionamiento en la Figura 3.150.

Figura 3.150: ejemplo de funcionamiento de CreateFilename

ii. DateToPxiFilename

Convierte la fecha actual en los nombres de los ficheros creados por el PXi, tanto el de los datos como el fichero de índices, ambos de extensión TDMS. La especificación de estos ficheros podrá ser usada para recogerlos vía FTP.

Icono: 

Representa la creación de una cadena de caracteres teniendo en cuenta el PXi.

Argumentos de entrada:
-Dates: array de cadenas de caracteres con fechas

Argumentos de salida:
-File Especifications: clúster con las especificaciones de los ficheros a recoger del Pxi por FTP.

El funcionamiento del SubVI se puede ver en la Figura 3.151.

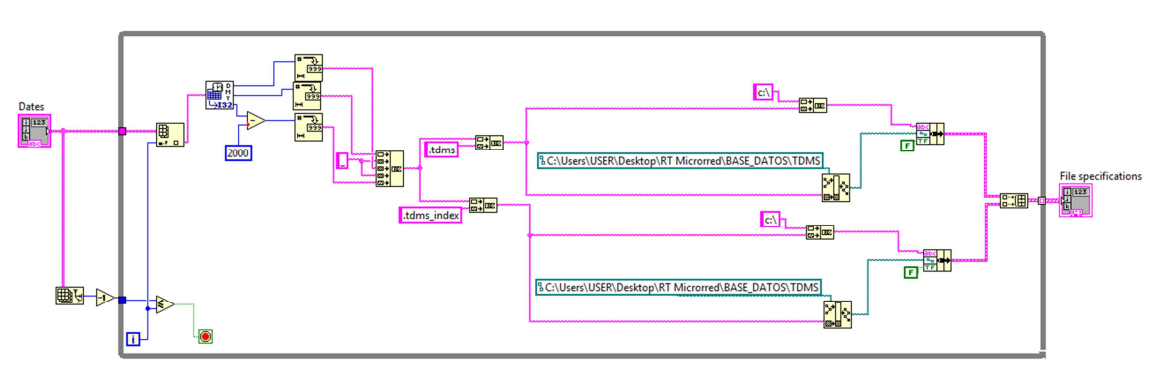

Figura 3.151: diagrama de bloques de DateToPxiFilename

Para cada una de las fechas especificadas en el array de entrada, se ejecuta una iteración del bucle.

Se usa la función creada TokenDateToInt para separar la fecha en número que correspondan al día, mes y año. Al año se le restan 2000 porque únicamente queremos las dos últimas cifras.

Se concatenan con las extensiones tdms y tdms_index, y cada una de éstas, por un lado con c:\, que será la ubicación del fichero en el PXi y por otro con C:\Users\USER\Desktop\RT Microrred\BASE_DATOS\TDMS, que será la ubicación en el ordenador actual. Uniéndolo todo en un clúster tendremos la especificación de ficheros adecuada.

Podemos ver un ejemplo de funcionamiento en la Figura 3.152.

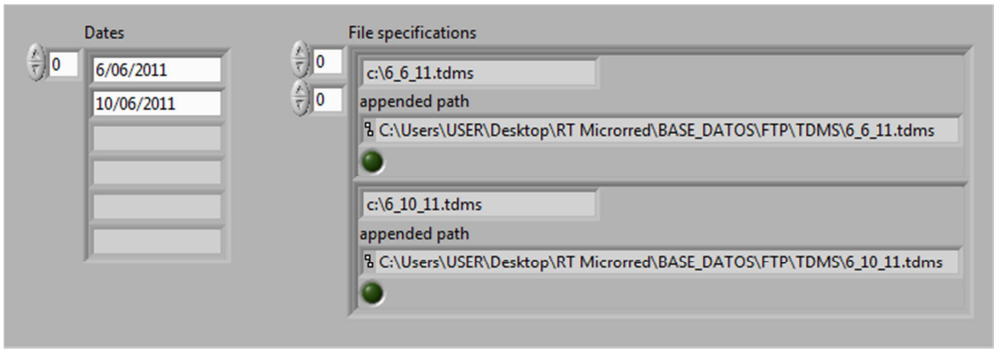

Figura 3.152: ejemplo de funcionamiento de DateToPxiFilename

iii. LastFileDate

Extrae la fecha de un fichero de configuración llamado date.txt, utilizado en RecolectarFTP (3.2.8). Si la hora no es la inmediata anterior a la actual, devuelve TRUE en un booleano.

Icono: 

Representa la hora almacenada en un fichero de texto.

Argumentos de entrada:
-Now: Timestamp con el momento actual.

Argumentos de salida:
-FileDate: cadena de caracteres con la fecha del fichero en formato dd/mm/yyyy
-Recoger datos: valdrá TRUE si la hora almacenada no corresponde con la inmediata anterior a la actual.

El funcionamiento del SubVI se puede ver en la Figura 3.153.

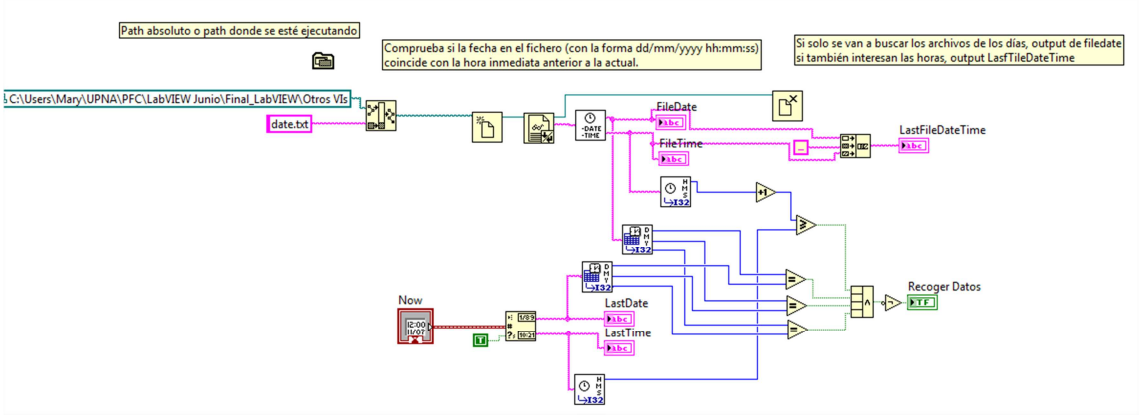

Figura 3.153: diagrama de bloques de LastFileDate

Primero, se abre el fichero y se extrae la fecha y hora en cadena de caracteres. Con la función creada TokenDateTime se separa por un lado la fecha, que será llevaba a la variable de salida, y por otro la hora.

Después se separa la hora actual y se usa la función creada TokenTime y la función creada TokenDate para separar en números enteros tanto las fechas y horas del fichero y actual. Se hacen comprobaciones para saber si la fecha es la misma y la hora es la inmediata anterior a la actual. Si todo esto se cumple, se enviará FALSE a la salida, si no, se enviará TRUE.

Podemos ver un ejemplo de funcionamiento en la Figura 3.154.

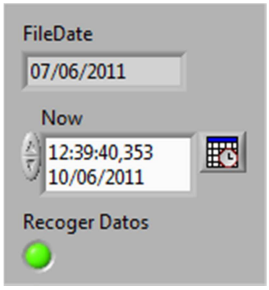

Figura 3.154: ejemplo de funcionamiento de LastFileDate

iv. SetTimeFile

Este VI almacena en un fichero de datos la hora deseada, sustituyendo la que estaba almacenada previamente.

Icono: 

Representa la asignación de la hora al fichero.

Argumentos de entrada:
-Now: fecha y hora actual en formato timestamp

No tiene argumentos de salida.

El funcionamiento del SubVI se puede ver en la Figura 3.155.

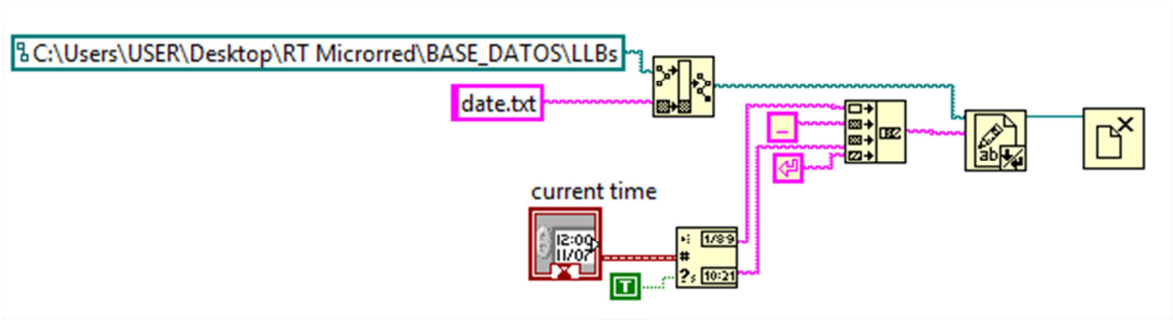

Figura 3.155: diagrama de bloques de SetTimeFile

Extrae las cadenas de caracteres con la fecha y la hora del momento actual y los escribe en el fichero de texto, para cerrarlo a continuación.

v. WriteToXLS

Escribe un conjunto de datos con sus fechas y horas a un fichero de MicrosoftExcel.

Icono: 

Representa la extracción de datos y su escritura en un fichero de celdas de Excel.

Argumentos de entrada:
-colSelected: array de cadenas de caracteres con los nombres de las variables de las que se tienen datos
-Time: array de timestamps con las fechas y horas de los datos
-Data: matriz con los datos en forma de números dobles

Argumentos de salida:
-XLS path: path que contiene el directorio de Microsoft Excel creado

El funcionamiento del SubVI se puede ver en la Figura 3.156.

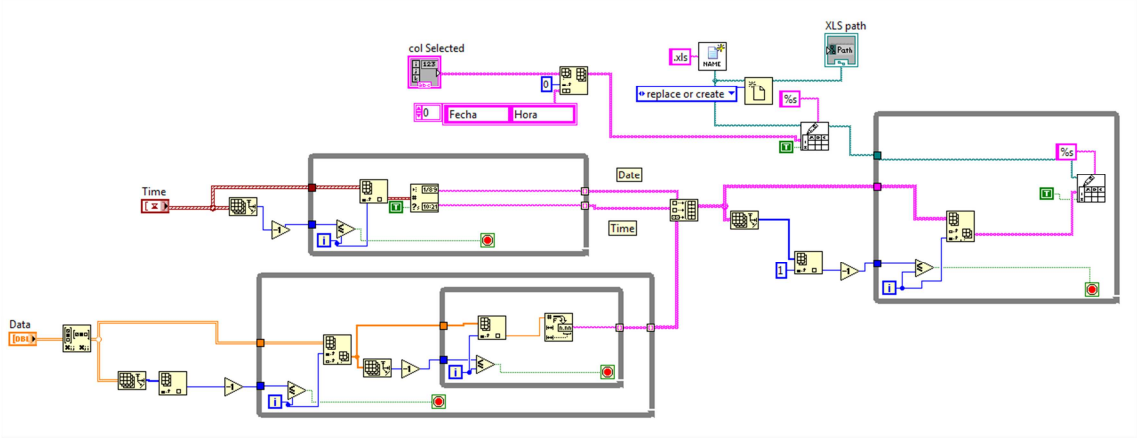

Figura 3.156: diagrama de bloques de WriteToXLS

Se tratan los distintos argumentos de entrada por separado.

-Al array con los nombres de las columnas seleccionadas se le inserta en las dos primeras posiciones la Fecha y Hora, ya que siempre tendremos esos datos.

-El array con los timestamps es convertido posición a posición en dos arrays de cadenas de caracteres con la fecha y la hora.

-La matriz con los datos también es convertida, en un doble bucle, a una matriz de cadenas de caracteres con los datos.

Todos los datos se unen construyendo un único array de datos.

Primero, se usa la función creada CreateFilename para crear el nombre del fichero de Microsoft Excel, y se escribe como primeros datos los nombres de las columnas.

A continuación, en un bucle, se van escribiendo las posiciones de los datos que ya se han tratado.

f) ChartToolsLLB

En esta librería se han agrupado los SubVIs que contribuyen a la representación gráfica de los datos, o que comprueban los mismos de cara a su representación.

i. CheckError

Este SubVI comprueba si dentro de los datos que se recogen, la variable de error de un módulo indica que se ha producido algún error, y devuelve TRUE en caso de que no haya sido así.

Icono: 

Representa la pregunta sobre si los datos son correctos o no.

Argumentos de entrada:
-DataPC: array de números dobles con los datos de la Microrred
-Met?(f)(opcional): booleano que indica si los datos a comprobar corresponden a las variables meteorológicas. Por defecto vale FALSE.
-MenuMet (opcional): indica el valor escogido para representar en las variables meteorológicas
-Índice: índice del dispositivo dentro de los datos

Argumentos de salida:
-Ok: valdrá TRUE si no existe ningún error, y FALSE en caso contrario.

El funcionamiento del SubVI se puede ver en la Figura 3.157.

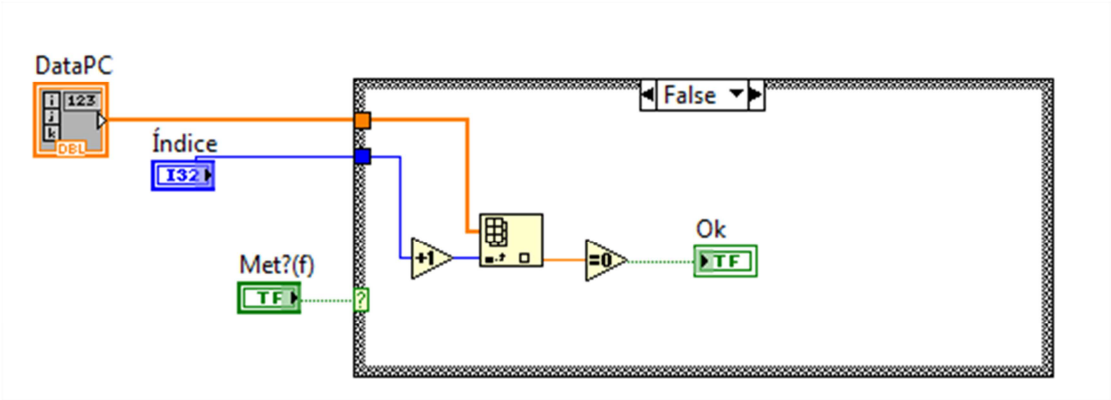

Figura 3.157: diagrama de bloques de CheckError para las variables no meteorológicas

En caso de que corresponde no corresponda a variables meteorológicas, la variable de error estará en una posición posterior al índice, por lo que, si esa posición vale 0, no habrá ningún error.

En el caso de las variables meteorológicas se considerará también el valor de MenuMet, que indica cuál de las variables meteorológicas queremos representar. Podemos ver en funcionamiento de este caso en la Figura 3.158.

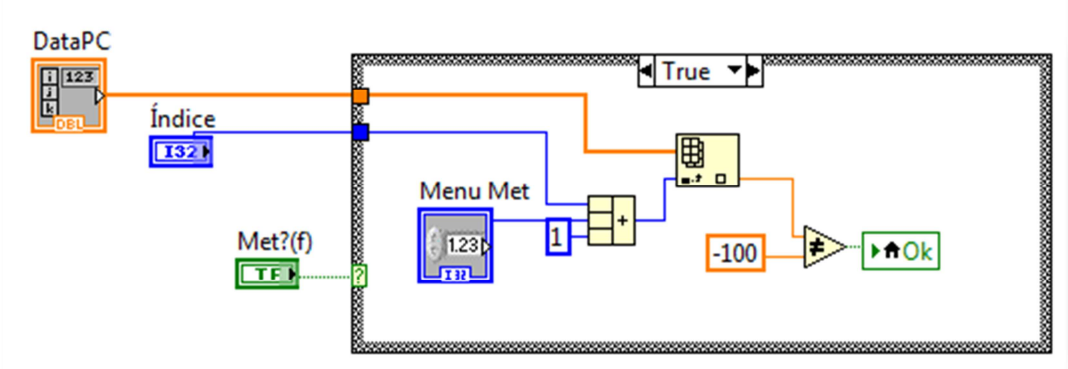

Figura 3.158: diagrama de bloques de CheckError para las variables meteorológicas

En este caso, habrá que acceder a la posición del dato exacto a representar y comprobar si no vale -100. En ese caso, no se habrá producido error y se devolverá TRUE.

ii. DrawPicture

Este SubVI se ha creado para representar imágenes dentro del panel frontal o interfaz con el usuario.

Icono: 

Representa el hecho de dibujar una imagen.

Argumentos de entrada:
-Path: ruta con la imagen a representar

Argumentos de salida:
-Picture: dibujo a representar en un cuadro de dibujo de LabVIEW (2D Picture Control)

El funcionamiento del SubVI se puede ver en la Figura 3.159.

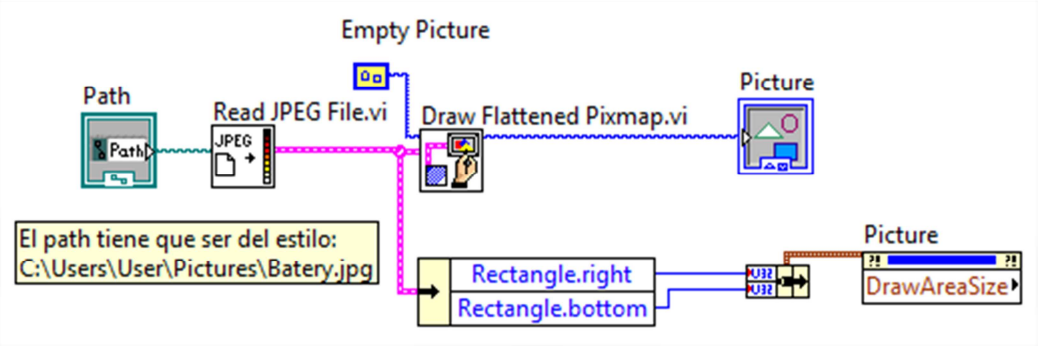

Figura 3.159: diagrama de bloques de DrawPicture

Las imágenes se encuentran en:
C:\Users\USER\Desktop\RT Microrred\BASE_DATOS\Imagenes
Ese path está almacenado en una variable Global llamada “Imágenes”. A partir del path se lee la imagen y se especifican sus características.

Podemos ver un ejemplo de funcionamiento en la Figura 3.160.

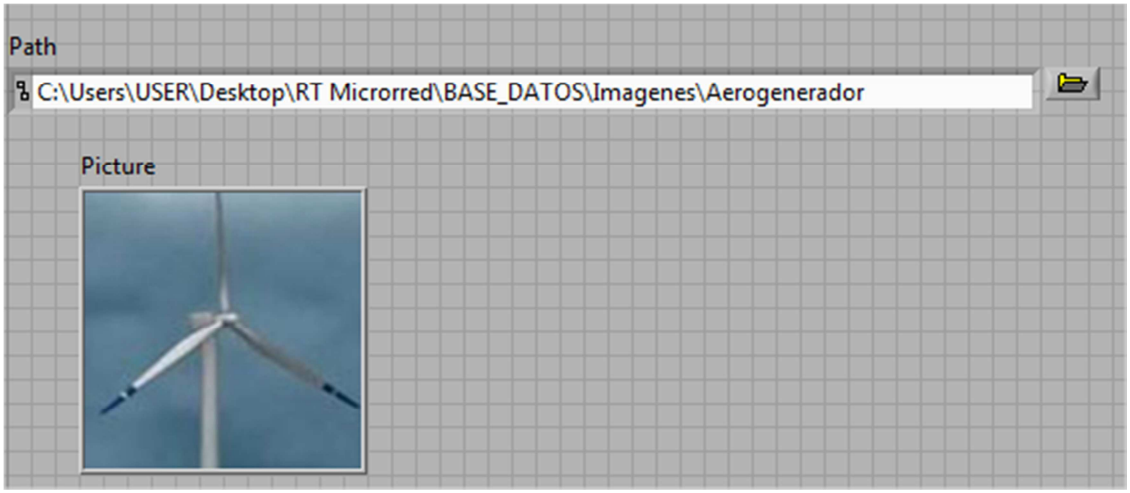

Figura 3.160: ejemplo de funcionamiento de DrawPicture

iii. MenuToChart

Este SubVI obtiene a partir del array con los datos de la Microrred (DataPC), del dispositivo requerido y la opción en el menú, el dato correspondiente.

Icono: 

Representa el paso desde el menú al dato, que se representará en una gráfica.

Argumentos de entrada:
-Menú: entero con la posición del menú seleccionada.
-Array: array de números dobles con los datos de la Microrred
-Index: índice correspondiente al módulo requerido.

Argumentos de salida:
-Result: número doble con el dato deseado.

El funcionamiento del SubVI se puede ver en la Figura 3.161.

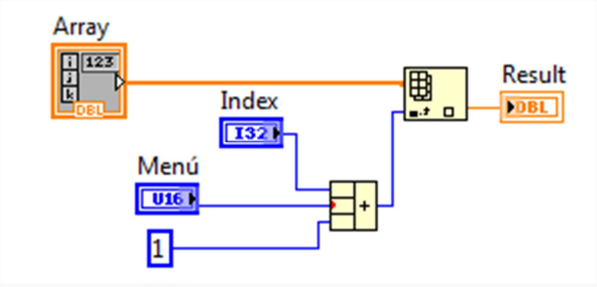

Figura 3.161: diagrama de bloques de MenuToChart

A partir del índice especificado, estará el código del módulo y después los datos.

En la posición [índice] estará el código.

En la posición [índice +1] estará el error.

En la posición [índice + 1 + menú] estará el valor buscado, siendo error la posición 0 en todos los casos menos híbrido inversor e híbrido eólica. En esos casos, como tampoco tienen código de error, el índice será 2 unidades menos que el índice real.

g) StringAndNumericTools

Librerías correspondientes al manejo de cadenas de caracteres y de números.

i. ArrayToString

Convierte un array de cadenas de caracteres en una única cadena en la que los elementos están separados por tabuladores. Además, inserta al principio los elementos Fecha y Hora.

Icono: 

Representa el paso desde array a cadena de caracteres.

Argumentos de entrada:
-ColSelected: array de cadenas de caracteres

Argumentos de salida:
-String: cadena con los elementos separados por tabuladores

El funcionamiento del SubVI se puede ver en la Figura 3.162.

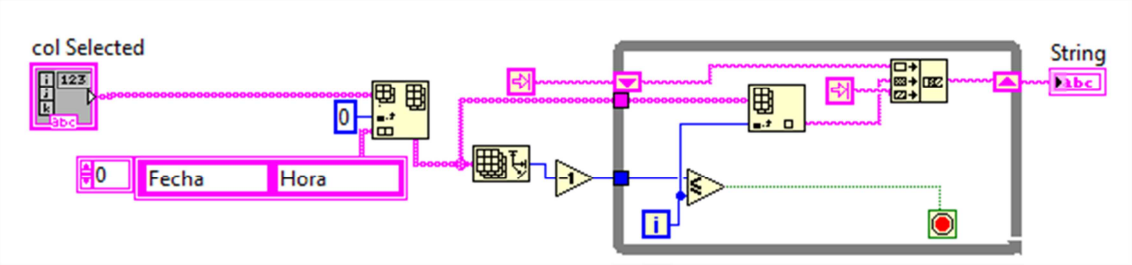

Figura 3.162: diagrama de bloques de ArrayToString

Primero añade al principio del array con los nombres la fecha y la hora. Después, para cada posición del array, concatena el elemento con un tabulador y el array del resultado, que se inicializa con un tabulador. Podemos ver un ejemplo de funcionamiento en la Figura 3.163.

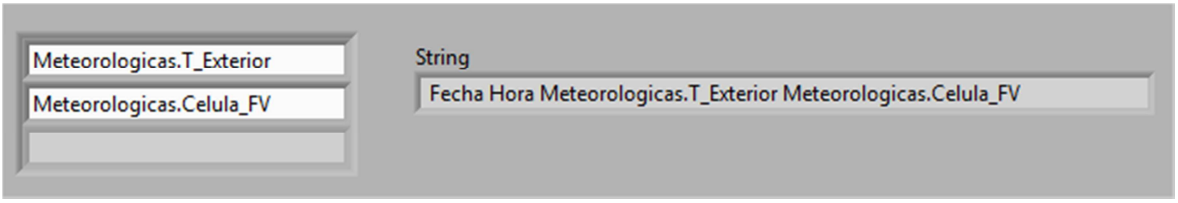

Figura 3.163: ejemplo de funcionamiento de ArrayToString

ii. DoubleToDateTime

Convierte un timestamp en forma de número doble a cadenas de caracteres con la fecha y la hora.

Icono: 

Representa la conversación de número a fecha y hora.

Argumentos de entrada:
-Numeric: número doble que representa una fecha y hora.

Argumentos de salida:
-Date: cadena de caracteres con la fecha en formato YYYY-MM-DD
-Time: cadena de caracteres con la hora en formato HH:MM:SS

El funcionamiento del SubVI se puede ver en la Figura 3.164.

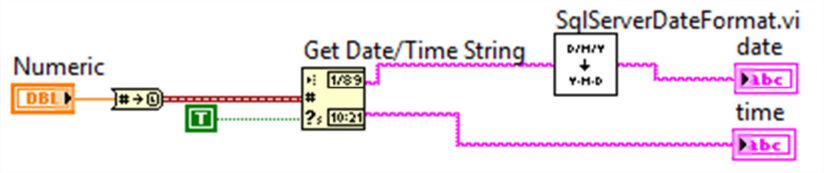

Figura 3.164: diagrama de bloques de DoubleToDateTime

Simplemente se realiza su conversión a timestamp, a continuación se extraen la fecha, que se convierte mediante la función creada SqlServerDateFormat, y la hora.

Podemos ver un ejemplo de funcionamiento en la Figura 3.165.

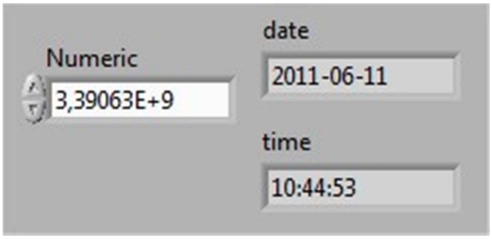

Figura 3.165: ejemplo de funcionamiento de DoubleToDateTime

iii. DoubleWithoutMilliseconds

Convierte un número doble que representa una fecha con la hora en formato HH:MM:SS'sss a una fecha con la hora con el formato HH:MM:SS .

Icono: 

Representa la conversión del número doble.

Argumentos de entrada:
-Double HH:MM:SS'sss: número doble con fecha y la hora en el formato HH:MM:SS'sss.

Argumentos de salida:
-Double HH:MM:SS: número doble con la fecha y la hora en formato HH:MM:SS.

El funcionamiento del SubVI se puede ver en la Figura 3.166.

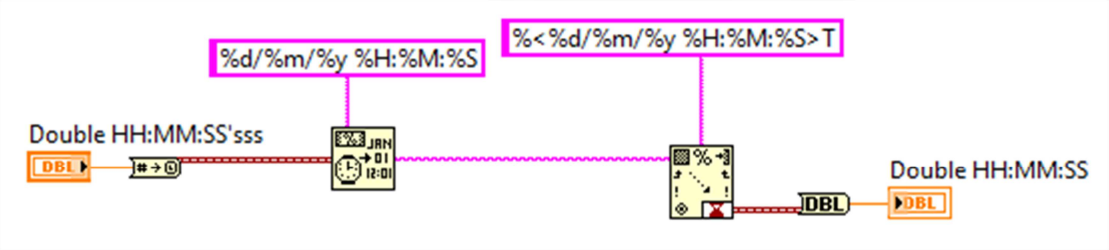

Figura 3.166: diagrama de bloques de DoubleWithoutMilliseconds

Se convierte el número doble a timestamp, éste a cadena de caracteres con el formato dd/mm/yyyy HH:MM:SS, esto a timestamp y por último a número doble.

iv. StringToTimestamp

Convierte una cadena de caracteres con la fecha y la hora a un timestamp.

Icono: 

Representa la conversión de una fecha y hora en cadena de caracteres a timestamp.

Argumentos de entrada:
-Fecha: cadena de caracteres con la fecha en formato dd/mm/yyyy
-Hora: cadena de caracteres con la hora en formato HH:MM:SS

Argumentos de salida:
-Timestamp: timestamp correspondiente a la fecha y hora de entrada.

El funcionamiento del SubVI se puede ver en la Figura 3.167.

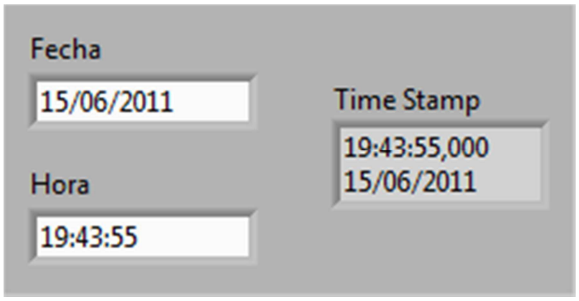

Figura 3.167: diagrama de bloques de StringToTimestamp

v. TimeStringToDouble

Convierte un string correspondiente a una fecha y hora a un número doble.

Icono: 

Representa la conversión desde la cadena de caracteres con la fecha y hora al número doble.

Argumentos de entrada:
-Time String: cadena de caracteres con la fecha y hora en formato dd/mm/yyyy HH:MM:SS.

Argumentos de salida:
-Time double: número doble con la fecha y la hora.

El funcionamiento del SubVI se puede ver en la Figura 3.168.

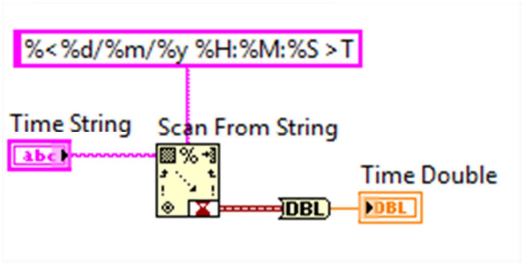

Figura 3.168: diagrama de bloques de TimeStringToDouble

vi. TimeToFormatString

Convierte un timestamp a dos cadenas de caracteres, con la fecha y la hora.

Icono: 

Representa la conversión desde el timestamp a la fecha y hora.

Argumentos de entrada:
-Timestamp: timestamp con una fecha y hora

Argumentos de salida:
-Date: cadena de caracteres con la fecha en formato DD/MM/YYYY
-Time: cadena de caracteres con la hora en formato HH:MM:SS.

El funcionamiento del SubVI se puede ver en la Figura 3.169.

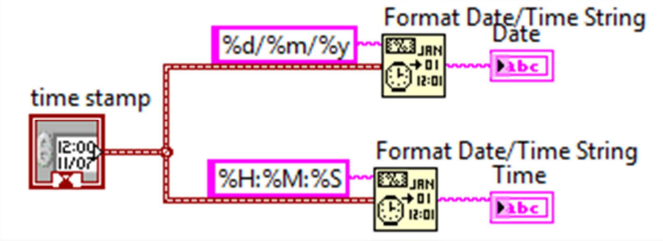

Figura 3.169: diagrama de bloques de TimeToFormatString

Simplemente se realiza la conversión adecuada con la función proporcionada por LabVIEW.

vii. TokenDateTime

Convierte una cadena de caracteres con la fecha y hora en dos cadenas de caracteres.

Icono: 

Representa la extracción de la fecha y hora por separado.

Argumentos de entrada:
-DateTime: cadena de caracteres con la fecha y hora en el formato dd/mm/yyyy HH:MM:SS.

Argumentos de salida:
-Date: cadena de caracteres con la fecha en formato dd/mm/yyyy.
-Time: cadena de caracteres con la hora en formato HH:MM:SS.

El funcionamiento del SubVI se puede ver en la Figura 3.170.

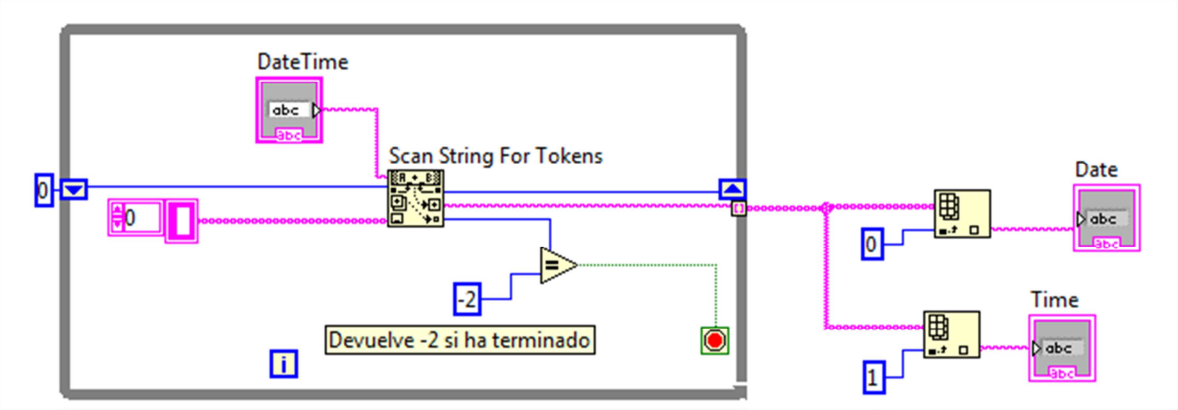

Figura 3.170: diagrama de bloques de TokenDateTime

Se hace uso de la función de LabVIEW ScanStringForTokens para separar la entrada considerando el espacio como separador. Se devuelve un array con dos posiciones, en el cuál la primera corresponde a la fecha y la segunda a la hora.

Podemos ver un ejemplo de funcionamiento en la Figura 3.171.

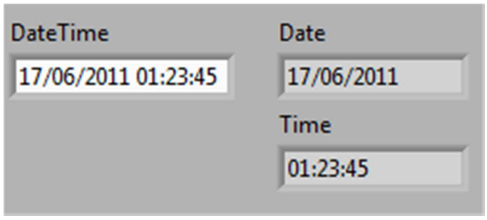

Figura 3.171: ejemplo de funcionamiento de TokenDateTime

viii. TokenDateToInt

Convierte una fecha de cadena de caracteres en formato dd/mm/yyyy a números enteros que representan el valor del día, el mes y el año.

Icono: 

Representa la conversión a números enteros.

Argumentos de entrada:
-Date: cadena de caracteres con una fecha en formato dd/mm/yyyy.

Argumentos de salida:
-Day: número entero que representa los días
-Month: número entero que representa el mes
-Year: número entero que representa el año

El funcionamiento del SubVI se puede ver en la Figura 3.172.

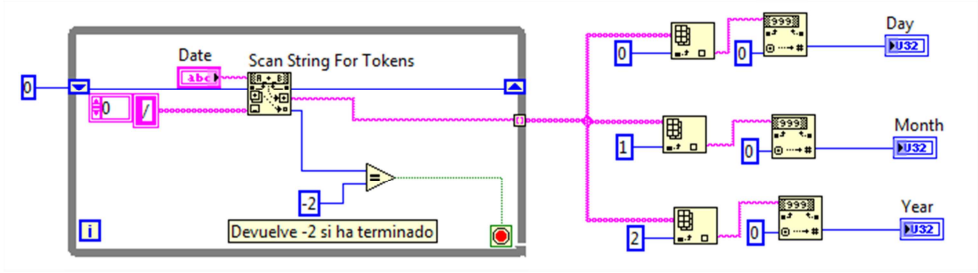

Figura 3.172: diagrama de bloques de TokenDateToInt

Hace uso de la función de LabIEW ScanStringForTokens para separar la fecha usando como delimitador el slash (/). Cada uno de los elementos se convierte a número entero.
Podemos ver un ejemplo de funcionamiento en la Figura 3.173.

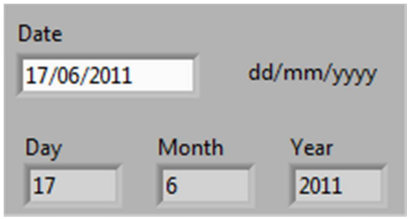

Figura 3.173: ejemplo de funcionamiento de TokenDateToInt

ix. TokenTime

Convierte una hora de cadena de caracteres en formato HH:MM:SS a números enteros que representan el valor de la hora, el mes y el segundo.

Icono: 

Representa la conversión a números enteros.

Argumentos de entrada:
-Hour: cadena de caracteres con una fecha en formato HH:MM:SS

Argumentos de salida:
-Hour: número entero que representa la hora
-Minute: número entero que representa los minutos
-Second: número entero que representa los segundos

El funcionamiento del SubVI se puede ver en la Figura 3.174.

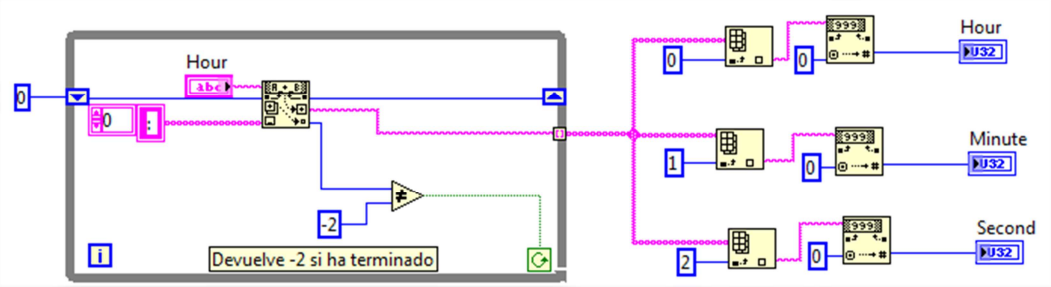

Figura 3.174: diagrama de bloques de TokenTime

Hace uso de la función de LabIEW ScanStringForTokens para separar la hora usando como delimitador los dos puntos (:). Cada uno de los elementos se convierte a número entero.
Podemos ver un ejemplo de funcionamiento en la Figura 3.175.

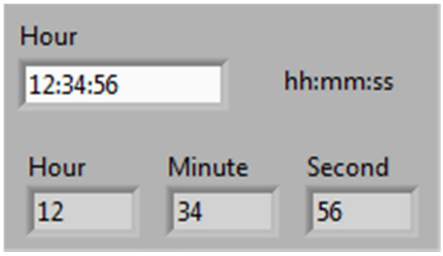

Figura 3.175: ejemplo de funcionamiento de TokenTime

4. Manual de usuario

ÍNDICE

4. Manual de usuario
4.1. Introducción
4.2. Arrancar y parar sistema
4.2.1. Arrancar sistema
4.2.2. Parar sistema
4.3. Cambios en la Microrred
4.3.1. Añadir módulo
4.3.2. Eliminar módulo
4.3.3. Añadir atributo a módulo
4.3.4. Eliminar atributo de módulo
4.4. Cambios en la Base de Datos
4.4.1. Información Base de Datos
4.4.2. Crear Base de Datos
4.4.3. Eliminar datos
4.4.4. Eliminar Base de Datos
4.4.5. Restaurar Base de Datos

4.1. Introducción

En este manual quedarán descritas las acciones que deben llevarse a cabo para los principales cambios que se deseen realizar en la Microrred, y que, por tanto, afecten al proyecto descrito. También se dedicará un apartado al manejo de la base de datos y su recuperación.

4.2. Arrancar y parar sistema

4.2.1. Arrancar sistema

Para arrancar el sistema hay que ejecutar una serie de programas de LabVIEW. Se encuentran dentro del proyecto de LabVIEW RT PXI.lvproj, cuya ubicación es:

C:\Users\USER\Desktop\RT Microrred\RT PXI.lvproj

Los diferentes programas ya se han añadido a ese proyecto, sin embargo, su código se puede encontrar en:

C:\Users\USER\Desktop\RT Microrred\BASE_DATOS\VisPantallas

Para iniciar un programa hacer doble clic sobre él dentro de la ventana del proyecto de LabVIEW y pulsar el botón Run (Control + R). Hay que iniciar, preferentemente en este orden, los siguientes programas:

- EsquemaComunicacion.vi
- RealTimeCharts.vi
- A elegir entre:

- EsquemaGeneralTermico.vi
- EsquemaGeneral.vi

- RecolectarFTP: Este programa, aunque no estaba dentro de las especificaciones principales, resulta muy útil que esté ejecutando. Su documentación se puede encontrar en la sección 3.2.8. Una vez iniciado, aparece una ventana emergente con 3 opciones:
  - Recoger datos desde fecha: con esta opción, se recoger los ficheros TDMS del Pxi a través del protocolo FTP, cuya fecha sea superior a la especificada en el fichero de texto date.txt que se encuentra en: C:\Users\USER\Desktop\RT Microrred\BASE_DATOS\LLBs En principio, este fichero tendrá la última fecha en la que el programa RecolectarFTP.vi se ha ejecutado con éxito, por lo que esta sería la opción normal de ejecución Si el PC se ha apagado y sabemos desde qué fecha se desean recuperar datos, podemos escribirla en el fichero, teniendo en cuenta que el formato es: dd/mm/yyyy HH:MM:SS
  - Iniciar control y ejecución: esta opción escribe en el fichero mencionado en el párrafo anterior la fecha y hora actual, y comienza su funcionamiento teniendo en cuenta ese dato. Por tanto, esta opción se utiliza la primera vez que se ejecuta el programa, o si se está seguro que no se han perdido datos dato que no ha estado apagado y simplemente el programa RecolectarFTP.vi no ha estado en funcionamiento. En caso de duda, es la opción recomendada.
  - Cancelar: sale del programa sin ejecutar nada más

Iniciar "HistoricosMicrorred.vi" únicamente cuando se deseen realizar gráficas con datos almacenados.

Iniciar "SelectData.vi" únicamente cuando se deseen generar informes o ejecutar programas externos sobre datos almacenados.

4.2.2. *Parar el sistema*

Para parar el sistema se deben detener los siguientes programas, siempre y cuando se estén ejecutando.

- EsquemaComunicación.vi
- RealTimeCharts.vi
- EsquemaGeneralTermico.vi
- EsquemaGeneral.vi
- RecolectarFTP.vi
- HistoricosMicrorred.vi
- SelectData.vi

4.3. **Cambios en la Microrred**

Uno de los requisitos principales por parte de los usuarios fue la flexibilidad de la aplicación, de modo que se pudieran añadir nuevos dispositivos a la Microrred, modificar uno ya existente para recibir mayor o menor cantidad de datos o eliminar alguno.

En este manual se hablará solo de la interacción con los datos y la base de datos, se da por supuesto la conexión del nuevo dispositivo con el PXi y la correcta configuración del mismo para el envío de los datos. Así, se parte de la base de que, una vez añadido, eliminado o modificado el dispositivo de la Microrred, se conoce la descripción exacta de los datos que se reciben en el PC. La descripción actual de los datos se puede consultar en la Figura 03 del apartado 3.1.2.

Los datos llegan al PC a través de la conexión con el PXi. Estos datos son representados en las distintas pantallas, así como insertados en la base de datos para su posterior consulta. Es por eso que un cambio en la Microrred supondrá cambios en todos los programas relacionados con los datos.

Durante el proceso de modificación es recomendable detener el proyecto de LabVIEW que se esté ejecutando sobre el PC. Lo que mayor importancia tiene es detener la inserción de los datos en la base de datos, ya que si ésta está en uso no permitirá modificaciones en su estructura. También se recomienda encarecidamente realizar una copia de seguridad de todo el proyecto antes de modificarlo.

Por otro lado, puede resultar conveniente seguir la nomenclatura de controles e indicadores, para que su posterior localización sea sencilla.

4.3.1. Añadir módulo

En este apartado se describen los pasos a seguir para añadir un nuevo módulo o dispositivo a la Microrred, por ejemplo, añadiendo otro Aerogenerador.

Hay que tener correctamente definida la especificación de los datos recibidos antes de comenzar, sabiendo qué posiciones ocupa cada módulo antes y después del cambio. Como recomendación, a la hora de añadir un nuevo módulo, añadir los datos al final de los ya definidos previamente, de modo que los cambios en los programas sean mínimos.

Los aspectos a modificar son los siguientes, y se recomienda hacerlo en este orden:

i. Base de Datos
ii. Esquema General
iii. Esquema General Térmico
iv. Esquema de Comunicación
v. Gráficas en Tiempo Real
vi. Históricos
vii. Selección de Datos

viii. FTP

Dirigirse a las secciones correspondientes dentro del Diseño de la Base de Datos (3.1) y del Diseño en LabVIEW (3.2) para ampliar la información sobre su funcionamiento interno.

i. Base de Datos

Añadir un nuevo módulo a la Microrred implica un nuevo grupo de datos que se desea almacenar. Por ello, tendremos que añadir una nueva relación a la base de datos, la cual almacene ese valor.

Para facilitar la interacción con el usuario se ha creado un interfaz en LabVIEW llamado MenuDB, y dentro de éste, en concreto, una opción de "Añadir módulo". La documentación de MenuDB se puede consultar en el apartado 3.2.9, mientras que la información concreta al programa que inserta la nueva relación aparece en la sección 3.2.10, dentro de la librería DBToolsLLB (CreateTable.vi).

Así pues, únicamente debemos ejecutar MenuDB y seleccionar la opción "Añadir módulo" antes de pulsa GO. Con esto aparecerá una ventana emergente que deberemos rellenar con el nombre de la tabla. Ponerle un nombre representativo a la nueva relación y recordarlo, pues es el que se usará para insertar los datos.

Además, se rellenará un array para cada uno de los datos que deseamos recoger sobre el nuevo módulo, como se muestra en la Figura 4.1. No se deben dejar posiciones del array desocupadas (por ejemplo, rellenando la posición 0 y la 2) ya que ello desembocará en error y no permitirá crear la relación en la base de datos. Siempre que sea un dato numérico seleccionaremos como tipo Single (SGL). Tener en cuenta que se debe desmarcar siempre el CheckBox "allow null?".

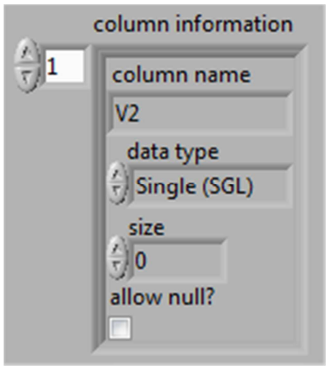

Figura 4.1: Definición de nuevas columnas en nueva relación

Al terminar de definir los atributos y pulsar GO, aparecerá una ventana emergente para confirmar la acción.

En cualquier momento se puede consultar la información disponible sobre la base de datos en MenuDB/Información Base de Datos.

ii. Esquema General (EsquemaGeneral.vi)

En este programa únicamente se muestran los datos, no se tratan de ninguna manera.

Lo primero, modificar el interfaz con el usuario al gusto de modo que se pueda añadir una nueva imagen y un nuevo display para representar al nuevo módulo. Se considera que la imagen se ha obtenido del tamaño deseado y se sabe su ubicación, que coincide con la ubicación del resto de imágenes, cuyo valor se puede consultar en Global.vi, dentro de la variable global Imágenes, que contiene el path. La imagen debe tener un formato jpg.

Vamos a suponer que deseamos añadir un nuevo Aerogenerador y lo queremos ubicar en el esquema cerca del ya existente. Necesitamos crear un lienzo en el que exponer nuestra imagen. Para ello acceder a Controls/Modern/Graph/Controls/2D Picture. Si hacemos clic derecho y seleccionamos propiedades podemos modificar su tamaño. Se recomienda comprobar el tamaño del resto de elementos y hacerlo similar, de forma que sea agradable visualmente para el usuario.

Con esto, se nos habrá creado en el diagrama de bloques un indicador Picture. Únicamente debemos emular el código para el resto de imágenes, como se puede ver en la Figura 4.2.

Figura 4.2: Asignación de imagen a Picture

Necesitamos unir la variable global Imágenes con la función de LabVIEW BuildPath, que nos unirá al path de la carpeta con las imágenes el nombre, en forma de cadena de caracteres, de nuestra nueva imagen. La salida de esa función se lleva a la función creada DrawPicture.vi, que se encuentra dentro de la librería ChartToolsLLB y cuyo funcionamiento se puede consultar en la sección 3.2.10. Como resultado obtenemos la imagen, que se lleva a nuestro indicador Picture, llamado P_Aerogenerador. Del mismo modo se crea la imagen que señalará el tipo de analizador de red conectado a nuestro módulo: PR300,…

Para añadir el letrero del nuevo módulo generamos un indicador de String desde Controls/Modern/String & Path / String Control. Dentro del diagrama de bloques, se habrá creado un indicador de String y haciendo clic derecho Create/Constant podemos elegir el texto a mostrar. Además, hay que configurar su tamaño y tipo de letra, por lo que generamos un Property Node desde clic derecho Create/Property Node con las características y los valores que se pueden ver en la Figura 4.3.

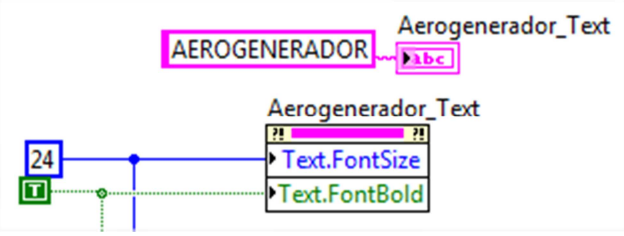

Figura 4.3: Configuración del letrero del nuevo módulo

En tercer lugar se añaden los leds que simulan las conexiones entre toda la Microrred. Para insertar un led acceder a Controls/Modern/Boolean/Square LED, y tener en cuenta que el ancho del mismo debe ser 16pxl.

Para añadir el indicador que mostrará el valor actual mostrado del nuevo módulo, insertamos un indicador numérico desde Controls/Modern/Numeric/Numeric Indicator. Dentro del diagrama de bloques se habrá creado un indicador numérico. Al igual que en el caso del letrero, tenemos que asignarle sus propiedades en un Property Node. Los valores a asignar se pueden ver en la Figura 4.4.

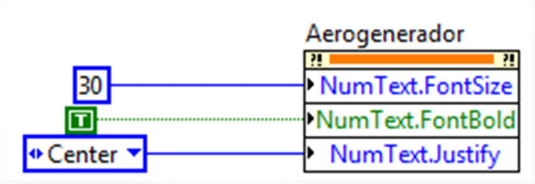

Figura 4.4: Configuración del indicador numérico del nuevo módulo

También se necesita un menú en caso de que el nuevo dispositivo suministre más de un dato, y poder elegir entre los disponibles. El menú se inserta desde Controls/Modern/Ring & Enum/Menu Ring. En el panel frontal hacemos clic derecho sobre el menú,accedemos a Propiedades y a Edit Items, donde especificamos las opciones del menú y su valor asociado. Escribir las opciones en el mismo orden en el que llegan los datos, es decir, si primero llega I y luego V, poner como primera opción I, que tendrá el valor asociado 0, y como segunda V. Al igual que en los elementos anteriores, creamos un Property Node con los valores de la Figura 4.5, que asociamos a nuestro nuevo menú.

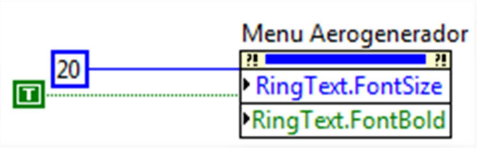

Figura 4.5: Configuración del menú del nuevo módulo

Con esto, se ha terminado con la configuración del panel frontal y resta la configuración del diagrama de bloques de modo que el indicador represente el valor correcto.

En el diagrama de bloques, cada módulo está asociado a una función creada que toma como argumento de entrada el índice de ese módulo en los datos. Si el nuevo módulo se ha añadido al final de los anteriores, únicamente se añadirá ése. En cambio, si añadir el

módulo ha cambiado el índice de alguno de los demás, habrá que modificar todos los correspondientes para que coincidan con su nuevo índice. Se puede ver la configuración de uno de los módulos en la Figura 4.6.

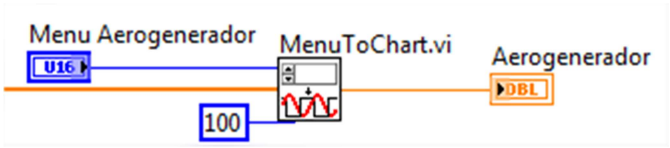

Figura 4.6: Configuración de los datos del indicador

La entrada naranja corresponde a DATAPC, los datos que provienen del PXi. El índice puede variar respecto a su valor real si los datos no coinciden con la estructura seguida por los demás módulos:
-Código del módulo (índice)
-Variable de error
-Variable1
-Variable2
-…

El valor del índice debe valer siempre 2 valores menos que la posición de la primera variable. Para comprobar el funcionamiento interno de la función creada MenuToChart consultar en la sección 3.2.10 la librería ChartToolsLLB.

iii. Esquema General Térmico (EsquemaGeneralTermico.vi)

Dado que el diagrama de bloques de este esquema es idéntico al del Esquema General, a la hora de modificarlo basta con seguir los pasos definidos para éste.

iv. Esquema de Comunicación (EsquemaComunicación.vi)

Para la modificación de imágenes y letreros, dirigirse al apartado referido al Esquema General. Tener en cuenta que los colores de los leds dependen del tipo de comunicación señalada en la leyenda del mismo diagrama.

De nuevo, tener en cuenta la posible modificación de los índices del resto de módulos al añadir el nuevo módulo a la Microrred.

Una vez conocido el índice del nuevo módulo y el número de datos del mismo, añadir un esquema como el de la Figura 4.7.

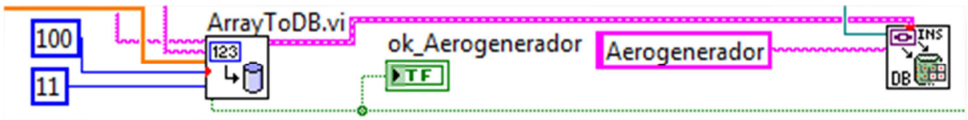

Figura 4.7: Configuración para la inserción de datos del nuevo módulo en la Base de Datos

En este caso, 100 representa al índice dentro del array de datos, que está representado por la línea naranja.El número 11 indica que este módulo tiene 5 datos, en cuya cuenta se incluye el código del módulo y el código de error, además de las variables.

Como se puede ver en la documentación de la función creada ArrayToDB (apartado 3.2.10, librería DBToolsLLB), tomará como entrada también Strings con la fecha y la hora actual. También obtenemos los datos (si tenemos 11 datos como entrada, la salida será en Data10) que insertamos en la función de LabVIEW destinada a ello, que encontramos en Functions/Connectivity/Database/DBTools Insert Data. Para ello, especificamos el nombre de la tabla en la base de datos, que hemos creado en el apartado i.

Como salida obtendremos un booleano que indicará si ha habido algún error, que corresponderá al led que une en el panel frontal ese módulo con el resto de la Microrred. Ese booleano lo llevaremos a la función booleana AND, que habrá que ampliar en un valor, y que no avisa de una posible alarma general de comunicación en la Microrred.

v. Gráficas en tiempo real (RealTimeCharts.vi)

A la hora de añadir una nueva gráfica hay que considerar si se desea añadir en una de las pestañas ya existentes o en una nueva. En caso de querer crear una nueva, basta con seleccionar la última y clic derecho / Add Page After, y renombrarla a nuestro gusto.

Para los indicadores de texto dirigirse a la sección de Esquema General.

Para crear una gráfica, insertar desde Controls/Modern/Graph/Waveform Chart. Considerar las mismas dimensiones que para las demás: 425x925 px.Hacer clic derecho en la gráfica y establecer en Chart History Length 7200, de manera que pueda almacenar 7200 datos, y como llega und dato por segundo, eso supone 2 horas. La configuración de la gráfica se puede ver en la Figura 4.8.

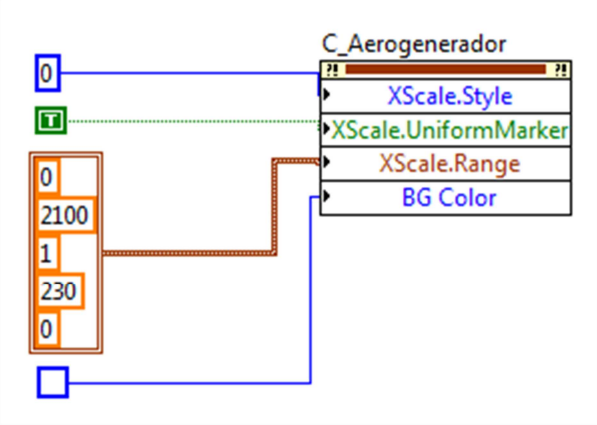

Figura 4.8: Configuración de gráfica en tiempo real

Para cada gráfica hay que añadir una serie de configuración como la que se ve en la Figura 4.9, que se puede consultar en el apartado 3.2.5.

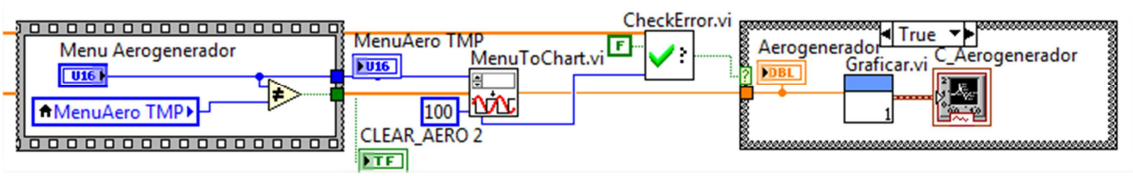

Figura 4.9: configuración diagrama de bloques nueva gráfica

Hay que tener en cuenta la posible modificación del resto de índices a la hora de asignar la variable de entrada en MenuToChart.vi.

Además, se crea un botón para limpiar la gráfica, del mismo estilo que los demás y con el mensaje CLEAR. Para ese botón y para la variable creada en la configuración anterior, que debería tener el mismo nombre acabado en 2, se añade la estructura case definida en la Figura 4.10.

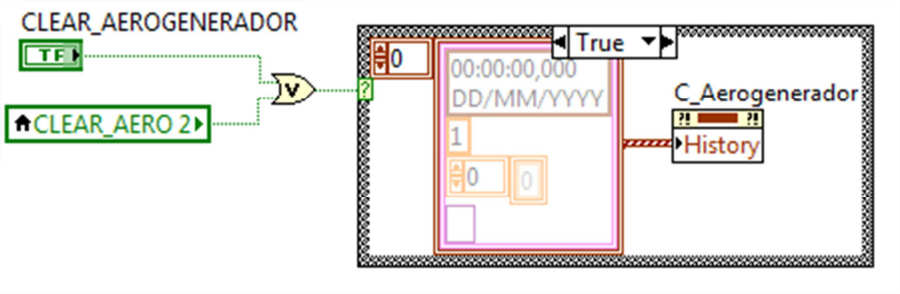

Figura 4.10: configuración borrado de gráfica

Dentro de la estructura case se encuentra un History, que podemos crearlo desde la gráfica, haciendo clic derecho y Create/Property Node/History Data. A continuación, hacer clic en el History Data recién creado y seleccionar Change All To Write.Una vez hecho, podemos hacer clic derecho y Create/Constant, que nos creará la estructura que limpie el historial de la gráfica.

vi. Históricos

Ya que almacenamos datos de un nuevo módulo, parece lógico que queramos también seleccionarlos.

En la parte del panel frontal de HistoricosMicrorred.vi únicamente debemos hacer cambios en la pestaña de Configuración, donde debemos insertar un letrero, al que aplicaremos la misma configuración que a los demás, y una serie de Checkbox (Controls/System /Boolean/ System Checkbox), correspondiendo uno a cada variable. Aunque se insertan los valores de error, únicamente se tienen en cuenta para seleccionar los datos y no se ha presentado como opción para escoger ya que no supondría de ningún interés representarlo.

Para cada uno de los checkbox se ha creado una variable booleana en el diagrama de bloques. Tenemos que tratarlos como se muestra en la Figura 4.11.

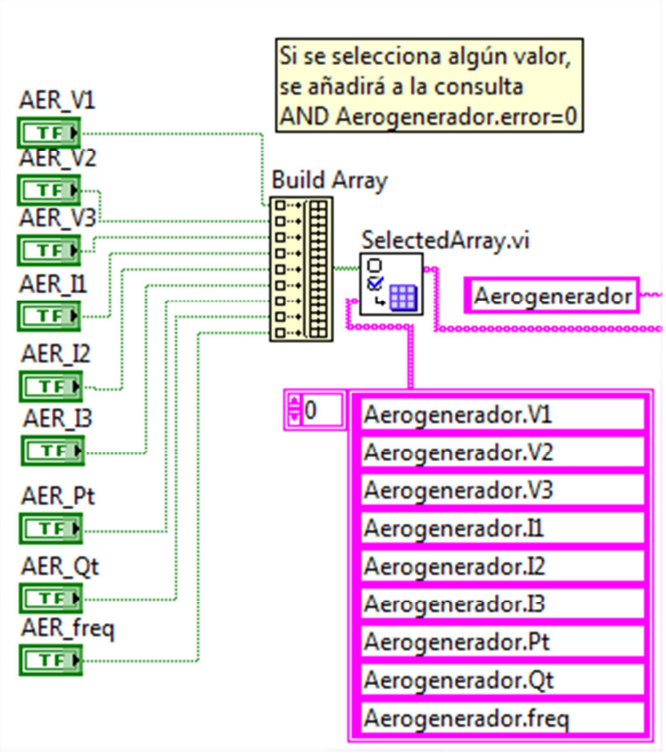

Figura 4.11: Tratamiento de los Checkbox del nuevo módulo

Se construye un array con ellos mediante la función de LabVIEW Build Array. Este array se introduce como argumento de entrada en la función creada SelectedArray (documentación en apartado 3.2.10, librería ArrayToolsLLB). Además, se construye una constante de tipo array (Functions/Programming/Array/Array Constant) en el que se introduce una constante de tipo String (Functions/Programming/String/String Constant) para convertirlo en un array de Strings. En cada una de las posiciones añadir el nombre del módulo y la variable separados por un punto, en el mismo orden en el que están los booleanos (Checkbox).

A continuación, añadir código como el mostrado en la Figura 4.12.

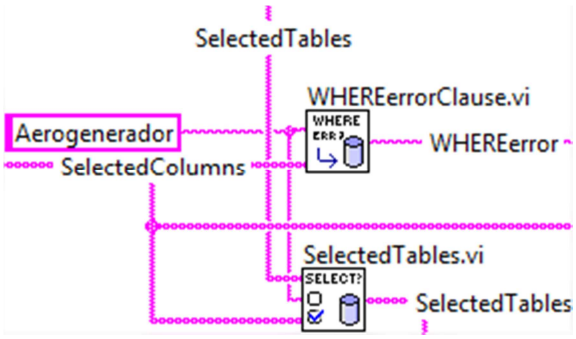

Figura 4.12: Tratamiento de variables seleccionadas

Se han etiquetado las uniones de forma que se conozcan las variables de entrada y salida de las funciones creadas (cuya documentación está en el apartado 3.2.10 dentro de

SQLToolsLLB). Si se añade al final del código, la entrada SelectedTables corresponderá a la salida del último módulo, y la salida irá a parar a las diversas funciones que requieren esta variables: SELECTclause, FROMclause, WHEREclause, ORDERBYclause.

Las columnas seleccionadas (SelectedColumns) de cada uno de los módulos, y en concreto de éste nuevo, se unen con la función de LabVIEW InsertIntoArray de modo que se obtengan todas las columnas seleccionadas (col Selected). Esta última variable servirá como argumento de entrada para SELECTclause y el bucle que rellena la leyenda de la gráfica anual.

La salida WHEREerror del nuevo módulo habrá que concatenarla a la sentencia SQL en la función de LabVIEW Concatenate Strings.

vii. Selección de Datos (SelectedData.vi)

Dado que el diagrama de bloques de este programa se basa en el de HistoricosMicrorred.vi, hay que realizar los mismos cambios que en éste.

viii. FTP (RecolectarFTP.vi)

Si se ha añadido un nuevo módulo se considera que los ficheros TDMS que se escriban con los datos han variado en tamaño, por lo que hay que realizar algún cambio.

Primero, en la función creada TDMStoDatabase (apartado 3.2.10, librería DBToolsLLB), en la entrada de la función ReshapeArray, que corresponde a la Figura 4.13.

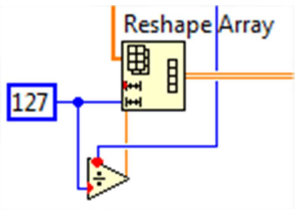

Figura 4.13: Entrada de Reshape Array

Hay que actualizar el indicador numérico, en este caso 127, al número total de datos, que será mayor que el actual. Para comprobar su corrección se aconseja ejecutar el SubVI por separado con un fichero TDMS actualizado con los nuevos datos y comprobar el indicador OutPutArray, de modo que en la primera posición se mantenga el doble con la fecha y la hora, y el resto de datos también estén en su lugar.

El siguiente SubVI a modificar es InsertDB (apartado 3.2.10, librería DBToolsLLB). Realmente, este SubVI se ha creado a partir del Esquema de Comunicación, por lo que los cambios que se realicen serán los mismos que en éste.

4.3.2. Eliminar módulo

En este apartado se describen los pasos a seguir para eliminar un módulo de los ya existentes en la Microrred, por ejemplo, si queremos desligar el Aerogenerador de la misma.

Hay que tener correctamente definida la especificación de los datos recibidos antes de comenzar, sabiendo qué posiciones ocupa cada módulo antes y después del cambio. Eliminar un módulo probablemente nos cambie las posiciones de los índices de todos los módulos cuyos datos llegaban después de él. Se llama índice de un módulo a la posición del array de datos (DataPC) en la que comienzan sus datos, que normalmente coincide con el código del módulo.

Los aspectos a modificar son los siguientes, y se recomienda hacerlo en este orden:

i. Base de Datos
ii. Esquema General
iii. Esquema General Térmico
iv. Esquema de Comunicación
v. Gráficas en Tiempo Real
vi. Históricos
vii. Selección de Datos
viii. FTP

Dirigirse a las secciones correspondientes dentro del Diseño de la Base de Datos (3.1) y del Diseño en LabVIEW (3.2) para ampliar la información sobre su funcionamiento interno.

i. Base de Datos

Eliminar un módulo de la Microrred implica que hay un grupo de datos que ya no se desea almacenar, por lo que tendremos que eliminar esa relación de la base de datos, junto con todos los datos que ha ido recogiendo.

Si no se desean almacenar más datos pero se desean conservar los recogidos hasta el momento puede omitirse este paso y continuar con el ii.

Para facilitar la interacción con el usuario se ha creado un interfaz en LabVIEW llamado MenuDB, y dentro de éste, en concreto, una opción de “Eliminar módulo”. La documentación de MenuDB se puede consultar en el apartado 3.2.9, mientras que la información concreta al programa que elimina los datos aparece en la sección 3.2.10, dentro de la librería DBToolsLLB (DropTableTable.vi).

Así pues, únicamente deberemos ejecutar MenuDB, seleccionar la opción “Eliminar módulo” y pulsar GO. Dentro de la nueva ventana emergente, esperamos a que se rellene el Menú con las relaciones existentes en la Base de Datos. Seleccionamos el módulo a eliminar y presionamos GO, como vemos en la Figura 4.14. Aparecerá una ventana

emergente para confirmar la acción. Con esto se eliminará la tabla, su relación con el resto de la base de datos y sus datos.

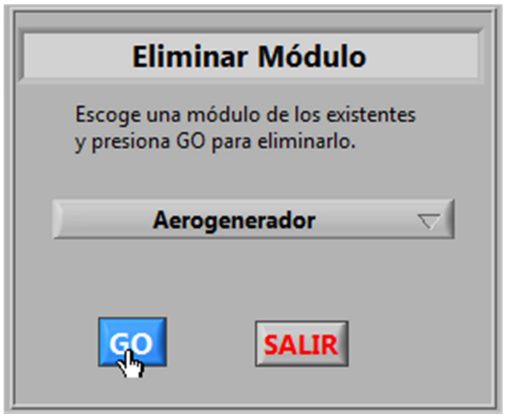

Figura 4.14: Eliminación de la relación Aerogenerador de la Base de Datos

En cualquier momento se puede consultar la información disponible sobre la base de datos en MenuDB/Información Base de Datos.

ii. Esquema General (EsquemaGeneral.vi)

En este programa únicamente se muestran los datos, no se tratan de ninguna manera.

En el diagrama de bloques quitar la sección de código correspondiente al módulo a eliminar. En caso de querer eliminar el Aerogenerador, eliminar el código de la Figura 4.15.

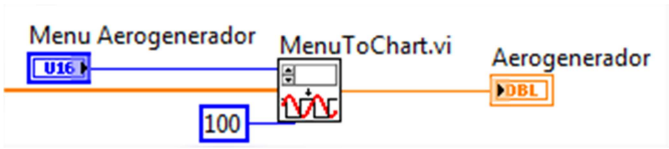

Figura 4.15: Código a borrar para eliminar el Aerogenerador del programa

Tener en cuenta, que, en este caso, el índice 100 correspondía al Aerogenerador, pero, al eliminarlo, esa posición corresponderá a datos de otro módulo, por tanto, tendremos que cambiar los índices de entrada a MenuToChart según la nueva distribución. Para comprobar el funcionamiento interno de la función creada MenuToChart consultar en la sección 3.2.10 la librería ChartToolsLLB. Con esto, hemos eliminado tanto el indicador numérico como el menú de selección del módulo.

Eliminar también el letrero que indicaba el nombre del módulo, cuyo código corresponde al mostrado en la Figura 4.16.

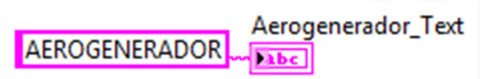

Figura 4.16: Código a borrar para eliminar el letrero del Aerogenerador

Para eliminar su imagen borrar el código de la Figura 4.17.

Figura 4.17: Código a borrar para eliminar la imagen del Aerogenerador del programa

De este modo se eliminará la imagen del Aerogenerador, pero seguirá mostrándose la imagen de su analizador de red. En el caso del Aerogenerador es un PR300 y haciendo doble clic en la imagen del Panel Frontal sabemos que su variable Picture asociada es la P_PR_300_1. Probablemente haya más de un módulo conectado a ese tipo de analizador de red, por lo tanto, únicamente eliminamos la variable P_PR_300_1, dejando las demás, como se ve en la Figura 4.18. En caso de que sólo existiera esa variable asociada a esa imagen, eliminar todo el código.

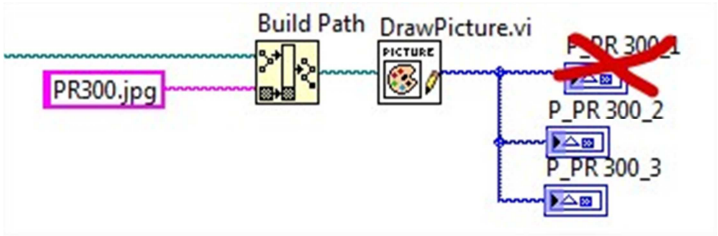

Figura 4.18: Código a borrar para eliminar la imagen del analizador de red del Aerogenerador

Por último, eliminar los leds del panel frontal que correspondían al conexionado eléctrico del Aerogenerador, y, con ellos, modificar la función AND que daba valor al booleano Alarma, ya que ahora tendrá menos entradas.

Al eliminar los controles se han eliminados sus configuraciones asociadas de tamaño de letra, color…(Property Node), sin embargo, puede que al eliminarse hayan quedado cables sueltos que impidan la compilación. Se pueden eliminar uno a uno o pulsar Ctrl + B para eliminar todos los cables sin conectar.

iii. Esquema General Térmico (EsquemaGeneralTermico.vi)

Dado que el diagrama de bloques de este esquema es idéntico al del Esquema General, a la hora de modificarlo basta con seguir los pasos definidos para éste.

iv. Esquema Comunicación (EsquemaComunicacion.vi)

Tener en cuenta la posible modificación de los índices del resto de módulos al eliminar el nuevo módulo de la Microrred, de modo que no basta con eliminar el código correspondiente a éste, si no que hay que modificar los índices de todos los módulos cuya posición de los datos en DataPC haya resultado modificada. En este VI, los índices aparecen como argumento de entrada en la función ArrayToDB, cuya documentación se puede consultar en el apartado 3.2.10 dentro de la librería DBToolsLLB.

Eliminar el código correspondiente a la Figura 4.19 para detener la inserción de valores, ya que la relación de la base de datos se ha eliminado.

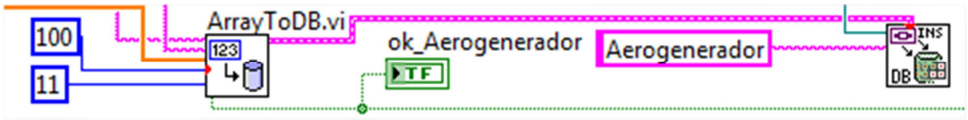

Figura 4.19: Código a borrar para detener la inserción de datos en la relación Aerogenerador

Para la eliminación de imágenes y letreros, dirigirse al apartado referido al Esquema General de esta misma sección. Se debe eliminar la imagen del módulo, la imagen de su indicador y los leds que indican el tipo de comunicación. Si eran los únicos leds de ese tipo, eliminar la opción correspondiente de la leyenda. Además, reducir las opciones de la función AND que establece el valor de la alarma, ya que se habrá eliminado el booleano correspondiente a ese módulo.

v. Gráficas en Tiempo Real (RealTimeCharts.vi)

A la hora de eliminar una gráfica de las ya existentes en el programa resulta recomendable eliminar primero su código asociado en el diagrama de bloques, ya que esto eliminará los controles del panel frontal.

Por tanto, suponiendo en todo momento que deseamos eliminar el módulo Aerogenerador, el primer código que debemos eliminar es el mostrado en la Figura 4.20.

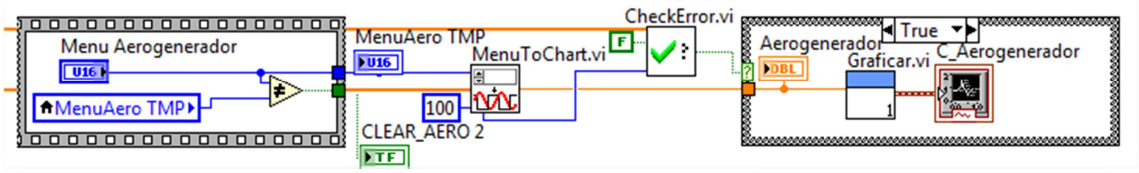

Figura 4.20: Código a borrar para eliminar la gráfica en tiempo real del Aerogenerador

El funcionamiento de este código se puede consultar en la sección 3.2.5 y el de las funciones utilizadas en la 3.2.10.

Con esto, se habrá eliminado tanto la gráfica como el menú de selección del dato deseado, así como sus configuraciones asociadas.

A continuación debemos eliminar el botón que limpia la gráfica, que corresponde al código de la Figura 4.21.

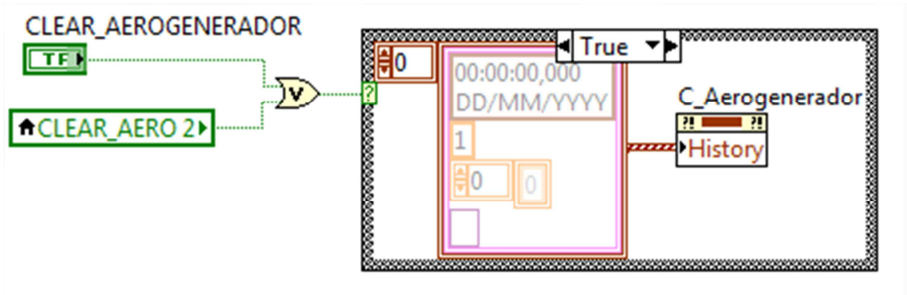

Figura 4.21: Código a borrar para eliminar el botón que limpia la gráfica de Aerogenerador.

Por último, debemos eliminar el letrero de la gráfica, cuyo código aparece en la Figura 4.22.

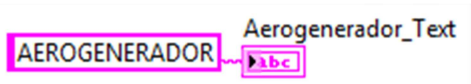

Figura 4.22: Código a borrar para eliminar el letrero de título de la gráfica de Aerogenerador

Una vez eliminada la gráfica y todas sus referencias, comprobar en el Panel Frontal si la página del TabControl correspondiente a esta gráfica ha quedado vacía. En ese caso, podemos eliminar la página haciendo clic derecho en ella y RemovePage.

Por último, reescribir los índices de todos los módulos. Los índices corresponden al número de entrada de la función MenuToChart y suele corresponder al código del módulo, o, si no se envía el código, corresponde a dos posiciones menos que la primera variable a controlar (omitiendo el código del módulo y el error). Así pues, modificar los índices de entrada de la función comentada de modo que todo funcione correctamente.

vi. Históricos (HistoricosMicrorred.vi)

Dado que dejamos de almacenar datos de un módulo no tiene sentido dar la opción de recuperarlos. En caso de que no hayamos eliminado los datos y los queramos conservar para consultarlos, se puede pasar a la siguiente sección. Sin embargo, tener en cuenta que las opciones como “Última hora”, “Último día”,… no devolverán ningún resultado, si no que tendremos que recuperar estos datos seleccionando el rango de tiempo adecuado desde “Entre dos fechas”.

Si queremos eliminarlo, debemos comenzar eliminando el código que nos permite seleccionar esos datos, que se puede ver en la Figura 4.23.

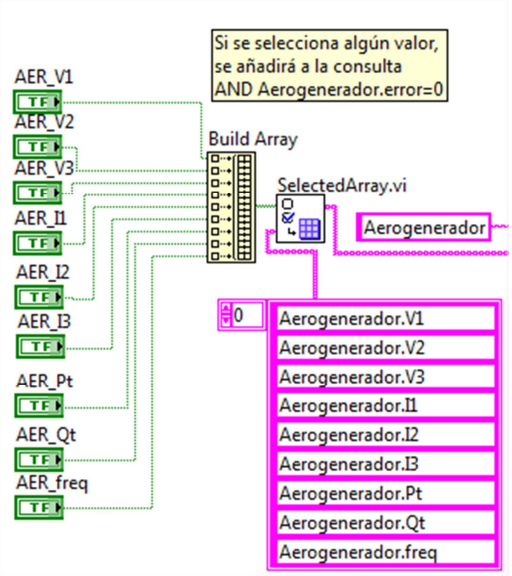

Figura 4.23: Código a borrar para eliminar la opción de seleccionar datos de Aerogenerador

Sin este código no podremos obtener las variables seleccionadas del módulo, y, por tanto, el código que calcule parte de la sentencia SQL correspondiente al módulo no tiene ninguna razón de ser. Así pues, eliminar el código mostrado en la Figura 4.24.

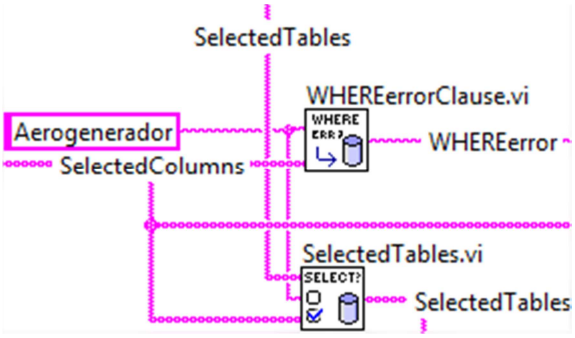

Figura 4.24: Funciones creadas que interactúan con los datos del módulo Aerogenerador

La variable SelectedTables comienza siendo un array vacío y es argumento de entrada y salida en la función SelectedTables correspondiente a todos los módulos, de forma que se va concatenando. Si el módulo a eliminar se encuentra en primer lugar, asignar la inicialización de SelectedTables (array de cadenas de caracteres vacío) al segundo de los módulo. Si se encuentra entre dos, asignar la salida de SelectedTables.vi del primero a la entrada de SelectedTables.vi del siguiente. Si se encuentra al final, llevar la salida de SelectedTables.vi del módulo anterior a todos los lugares correspondientes: SELECTclause.vi, FROMclause, WHEREclause, GROUPBYclause.

Se ha eliminado la salida SelectedColumns perteneciente a ese módulo, así que se debe eliminar una entrada de la función BuildArray que recoge las variables SelectedColumns de todos los módulos. Así, si no llega esta variable, el array correspondiente tendrá una entrada menos que habrá que reducir.

Además, se elimina la variable WHEREerror de este módulo, por lo tanto, la función ConcatenateStrings recibe una entrada menos, que habrá que reducir.

vii. Selección de Datos (SelectedData.vi)

Dado que el diagrama de bloques de este programa se basa en el de HistoricosMicrorred.vi, hay que realizar los mismos cambios que en éste.

viii. FTP (RecolectarFTP.vi)

Si se ha eliminado un módulo se considera que los ficheros TDMS que se escriban con los datos han variado en tamaño, por lo que hay que realizar algún cambio.

Primero, en la función creada TDMStoDatabase (apartado 3.2.10, librería DBToolsLLB), en la entrada de la función ReshapeArray, que corresponde a la Figura 4.25.

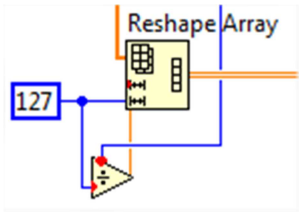

Figura 4.25: Entrada de Reshape Array

Hay que actualizar el indicador numérico, en este caso 127, al número total de datos, que será menor que el actual. Para comprobar su corrección se aconseja ejecutar el SubVI por separado con un fichero TDMS actualizado con los nuevos datos y comprobar el indicador OutPutArray, de modo que en la primera posición se mantenga el doble con la fecha y la hora, y el resto de datos también estén en su lugar.

El siguiente SubVI a modificar es InsertDB (apartado 3.2.10, librería DBToolsLLB). Realmente, este SubVI se ha creado a partir del Esquema de Comunicación, por lo que los cambios que se realicen serán los mismos que en éste.

4.3.3. Añadir atributo a módulo

En este apartado se describen los pasos a seguir para añadir un nuevo atributo a un módulo ya existente, por ejemplo, si del Aerogenerador queremos almacenar, además de todo lo actual, otro dato.

Hay que tener correctamente definida la especificación de los datos recibidos antes de comenzar, sabiendo qué posiciones ocupa cada módulo antes y después del cambio. Al añadir un atributo a un módulo, es lógico que los índices de los módulos cuyos datos se envían después de él queden incrementados en una unidad.

Los aspectos a modificar son los siguientes, y se recomienda hacerlo en este orden:

i. Base de Datos
ii. Esquema General
iii. Esquema General Térmico
iv. Esquema de Comunicación
v. Gráficas en Tiempo Real
vi. Históricos
vii. Selección de Datos
viii. FTP

Dirigirse a las secciones correspondientes dentro del Diseño de la Base de Datos (3.1) y del Diseño en LabVIEW (3.2) para ampliar la información sobre su funcionamiento interno.

i. Base de Datos

Añadir un atributo a un módulo ya existente en la Microrred implica cambiar la estructura de esa relación en la base de datos, de modo que tenga capacidad para albergan un dato nuevo.

Para facilitar la interacción con el usuario se ha creado un interfaz en LabVIEW llamado MenuDB, y dentro de éste, en concreto, una opción de “Añadir atributo”. La documentación de MenuDB se puede consultar en el apartado 3.2.9, mientras que la información concreta al programa que inserta la nueva relación aparece en la sección 3.2.10, dentro de la librería DBToolsLLB (AddColumn.vi).

Así pues, debemos ejecutar MenuDB y seleccionar del menú la opción “Añadir atributo”, con lo que aparecerá una nueva ventana emergente. Dentro de esta ventana, esperaremos a que “Menú Módulos” se pueble con los datos de las relaciones alojadas en la base de datos.

Debemos escribir el nombre del nuevo atributo y elegir del segundo menú el tipo de dato. Siempre que sea numérico elegiremos tipo Real. Por defecto, el tipo String tiene un tamaño de 50 caracteres. Si se desea configurar otro tamaño se recomienda usar el interfaz de SQL Server para crear la relación o para modificarla si la hemos creado desde el interfaz de LabVIEW.

Una vez rellenados todos los datos, al pulsar GO, aparecerá una ventana emergente que nos pedirá confirmar la operación.

Podemos ver un ejemplo de cómo añadir un atributo en la Figura 4.26.

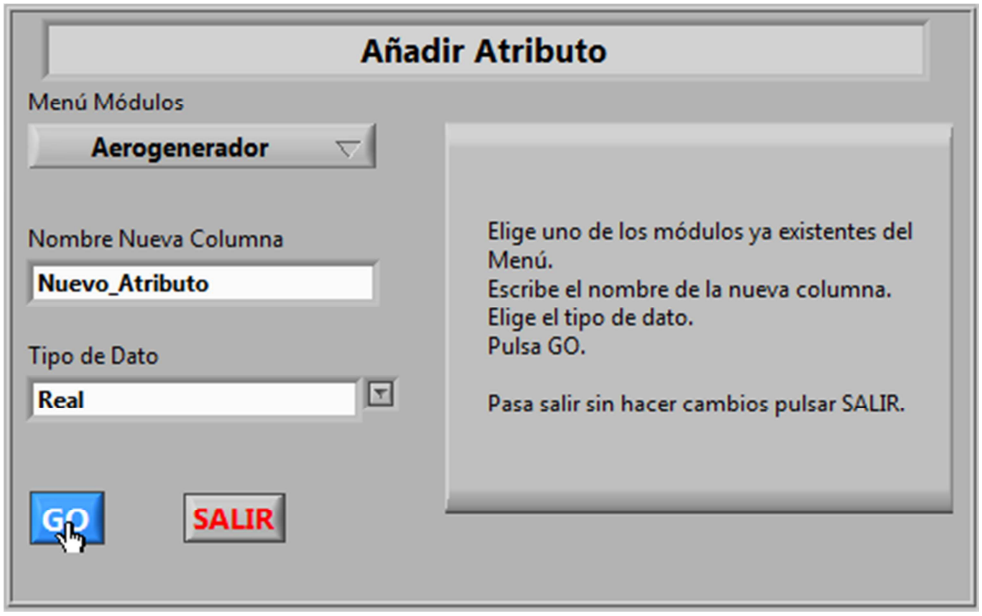

Figura 4.26: Añadir nuevo atributo a módulo Aerogenerador.

En cualquier momento se puede consultar la información disponible sobre la base de datos en MenuDB/Información Base de Datos.

ii. Esquema General (EsquemaGeneral.vi)

Si tratamos de añadir un nuevo dato a un módulo ya existente no hay que realizar grandes cambios en el panel frontal.

Deberemos acceder al menú del módulo en concreto, haciendo clic derecho y seleccionando Properties. Dentro de las propiedades debemos acceder a Edit Items y añadir el nuevo dato en el lugar correspondiente. Es decir, si se envía al final de todos los anteriores, añadir el nuevo elemento al final de los demás, como se puede ver en la Figura 4.27.

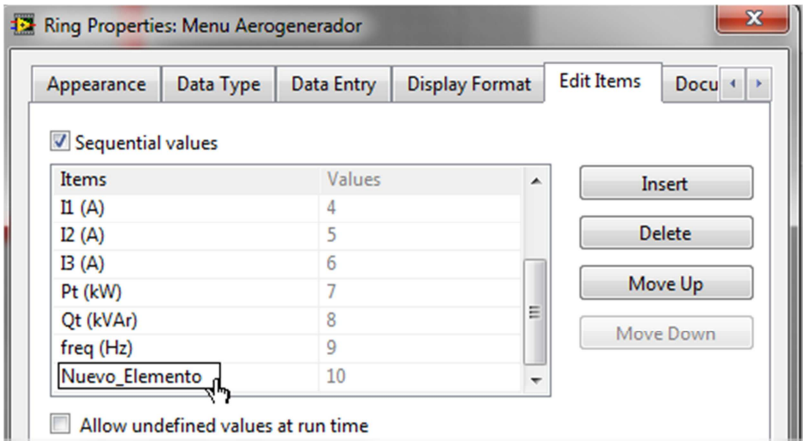

Figura 4.27: Añadir nuevo elemento a Menú Aerogenerador

El índice dentro del array de datos (DataPC) de nuestro módulo no habrá cambiado, porque sus datos seguirán comenzando en la misma posición. Sin embargo, excepto en la

ocasión en la que el módulo a modificar sea el último en la posición de los datos, el índice de todos cuyos datos vengan después se habrá desplazado una posición.

Es decir, si Aerogenerador comienza en la posición 100 y contiene 10 datos, además del código del módulo, el siguiente comenzará en la posición 111. Sin embargo, si ahora Aerogenerador tiene 11 datos, el siguiente tendrá que comenzar en el 112, y así con todos los que estén después.

El valor del índice debe valer siempre 2 valores menos que la posición de la primera variable. Para comprobar el funcionamiento interno de la función creada MenuToChart consultar en la sección 3.2.10 la librería ChartToolsLLB.

Sin embargo, excepto la variación de los índices en los módulos, añadir un nuevo atributo no produce ningún otro cambio en el Esquema General.

iii. Esquema General Térmico (EsquemaGeneralTermico.vi)

Dado que el diagrama de bloques de este esquema es idéntico al del Esquema General, a la hora de modificarlo basta con seguir los pasos definidos para éste.

iv. Esquema Comunicación (EsquemaComunicacion.vi)

En el esquema de comunicación tenemos que variar la cantidad de datos que se insertarán en la relación de la base de datos correspondiente el módulo tratado, ya que ahora alojaremos un atributo más. Es por eso que tenemos que modificar el código correspondiente a la Figura 4.28.

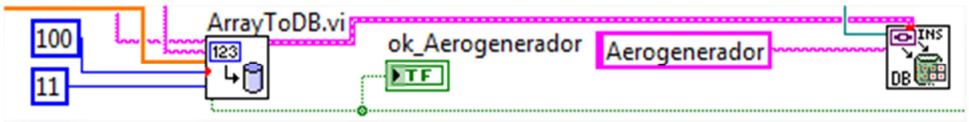

Figura 4.28: Código a modificar al añadir nuevo atributo

En este caso se trata del módulo Aerogenerador y su índice anterior valía 100 y su número de datos era 11 (código del módulo en la posición 100 y luego 10 datos). Si hemos añadido un nuevo atributo, la posición del código no variará pero si lo hará la cantidad de datos. Es decir, ahora el número de datos será 12 en lugar de 11. Y antes la salida era por Data10, pero ahora, al ser el número de datos 12 la salida se dará por Data11.

Como se explicaba en el Esquema General de esta misma sección, habrá que cambiar los índices de los módulos cuyos datos se recojan después del módulo que hayamos modificado. A la hora de modificar los índices, seguir la norma general por la que el valor del índice debe valer siempre 2 valores menos que la posición de la primera variable.

Excepto los valores de los índices, no hay que realizar ningún otro cambio en el Esquema de Comunicación para que funcione correctamente con la nueva configuración de la Microrred.

v. Gráficas en Tiempo Real (RealTimeCharts)

En la gráfica en tiempo real del módulo que estamos modificando debemos poder mostrar el nuevo dato que recogemos. Para ello, primero debemos modificar el menú en el que se muestran las posibilidades como se ha explicado en el Esquema General de este mismo apartado.

Además, como en todos los demás programas, hay que modificar el índice de los módulos cuyos datos se envían después del modificado, ya que si no la función MenuToChart no devolverá los resultados correctos. Como norma general, el valor del índice debe valer siempre 2 valores menos que la posición de la primera variable. Para comprobar el funcionamiento interno de la función creada MenuToChart consultar en la sección 3.2.10 la librería ChartToolsLLB.

vi. Históricos (HistoricosMicrorred.vi)

Debemos presentar la opción de recoger los datos de todas las variables del módulo, incluyendo el nuevo atributo añadido. Primero,en el panel frontal, dentro de la pestaña de Configuración, debemos añadir un nuevo CheckBox que represente el nuevo atributo.Crear un Property Node para el CeckBox del nuevo atributo con la configuración de la Figura 4.29.

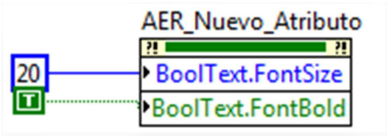

Figura 2.29: Configuración nuevo CheckBox

Además, hay que unir el nuevo CheckBox al array de booleanos que formaban el resto de variables del Módulo. Junto a esto añadir al array de cadenas de caracteres una posición con el nombre del módulo seguido de un punto y el nombre del atributo creado, como se ve en la Figura 4.30.

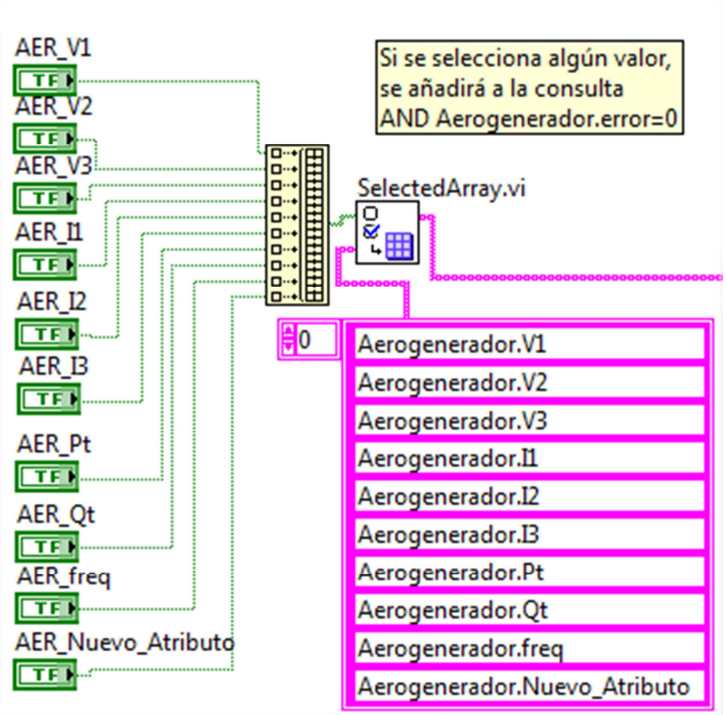

Figura 4.30: Configuración Nuevo Atributo en Aerogenerador

vii. Selección de Datos (SelectedData.vi)

Dado que el diagrama de bloques de este programa se basa en el de HistoricosMicrorred.vi, hay que realizar los mismos cambios que en éste.

viii. FTP (RecolectarFTP.vi)

Si se ha añadido un nuevo atributo a un módulo ya existente se considera que los ficheros TDMS que se escriban con los datos han variado en tamaño, por lo que hay que realizar algún cambio.

Primero, en la función creada TDMStoDatabase (apartado 3.2.10, librería DBToolsLLB), en la entrada de la función ReshapeArray, que corresponde a la Figura 4.31.

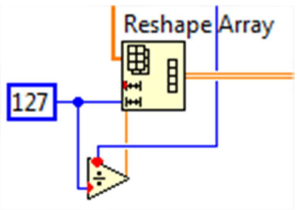

Figura 4.31: Entrada de Reshape Array

Hay que actualizar el indicador numérico, en este caso 127, al número total de datos, que será uno mayor que el actual. Para comprobar su corrección se aconseja ejecutar el SubVI por separado con un fichero TDMS actualizado con los nuevos datos y comprobar el indicador OutPutArray, de modo que en la primera posición se mantenga el doble con la fecha y la hora, y el resto de datos también estén en su lugar.

El siguiente SubVI a modificar es InsertDB (apartado 3.2.10, librería DBToolsLLB). Realmente, este SubVI se ha creado a partir del Esquema de Comunicación, por lo que los cambios que se realicen serán los mismos que en éste.

4.3.4. Eliminar atributo de módulo

En este apartado se describen los pasos a seguir para eliminar un atributo de un módulo ya existente, por ejemplo, si del Aerogenerador ya no queremos almacenar la variable freq.

Hay que tener correctamente definida la especificación de los datos recibidos antes de comenzar, sabiendo qué posiciones ocupa cada módulo antes y después del cambio. Al eliminar un atributo de un módulo, es lógico que los índices de los módulos cuyos datos se envían después de él queden decrementados en una unidad.

Los aspectos a modificar son los siguientes, y se recomienda hacerlo en este orden:

i. Base de Datos
ii. Esquema General
iii. Esquema General Térmico
iv. Esquema de Comunicación
v. Gráficas en Tiempo Real
vi. Históricos
vii. Selección de Datos
viii. FTP

Dirigirse a las secciones correspondientes dentro del Diseño de la Base de Datos (3.1) y del Diseño en LabVIEW (3.2) para ampliar la información sobre su funcionamiento interno.

i. Base de Datos

Eliminar un atributo de un módulo ya existente en la Microrred implica cambiar la estructura de esa relación en la base de datos, de modo que ya no tenga capacidad para albergan ese dato, y se eliminen los datos almacenados hasta ese momento de ese atributo.

Para facilitar la interacción con el usuario se ha creado un interfaz en LabVIEW llamado MenuDB, y dentro de éste, en concreto, una opción de “Eliminar atributo”. La documentación de MenuDB se puede consultar en el apartado 3.2.9, mientras que la información concreta al programa que inserta la nueva relación aparece en la sección 3.2.10, dentro de la librería DBToolsLLB (DropColumn.vi).

Así pues, debemos ejecutar MenuDB y seleccionar del menú la opción “Eliminar atributo”, con lo que aparecerá una nueva ventana emergente. Dentro de esta ventana, esperaremos a que “Menú Módulos” se pueble con los datos de las relaciones alojadas en la base de datos.

Una vez seleccionado el módulo sobre el que se eliminará un atributo pulsar en el botón “Consultar Atributos”, de modo que el segundo menú se rellene con los atributos pertenecientes a ese módulo. Aunque salgan como opción, no es aconsejable ni está permitido borrar los atributos de Fecha y Hora, ya que forman parte de la clave primaria de la relación, por lo que son necesarios para la identificación de los datos.

Una vez elegido del segundo menú el atributo a eliminar pulsar GO, como se muestra en la Figura 4.32.

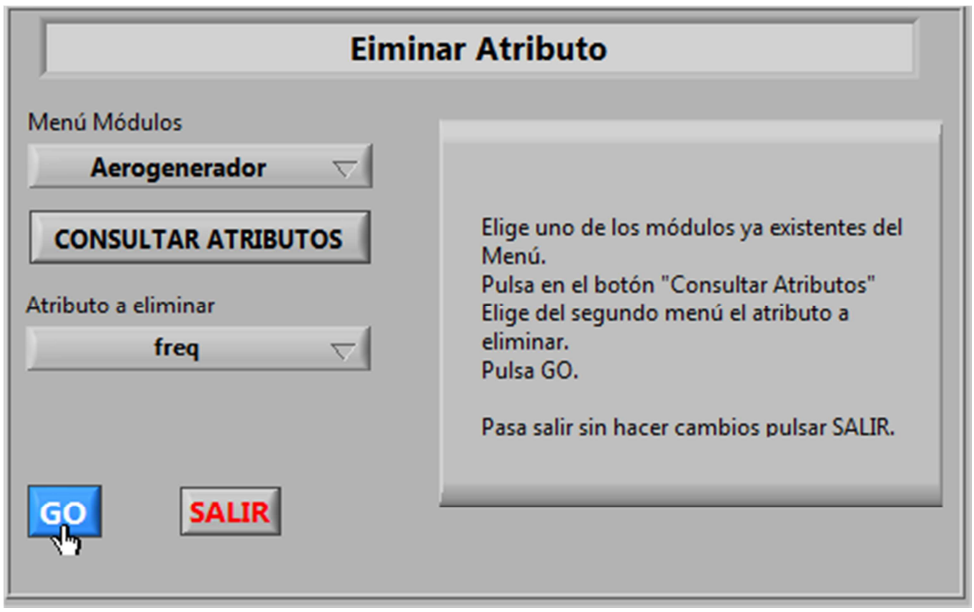

Figura 4.32: Eliminar un atributo de un módulo ya existente

Aparecerá una ventana emergente que nos pedirá confirmar la operación.

ii. Esquema General (EsquemaGeneral.vi)

Si tratamos de eliminar un dato de un módulo ya existente no hay que realizar grandes cambios en el panel frontal.

Deberemos acceder al menú del módulo en concreto, haciendo clic derecho y seleccionando Properties. Dentro de las propiedades debemos acceder a Edit Items y eliminar el dato correspondiente, como se ve en la Figura 4.33.

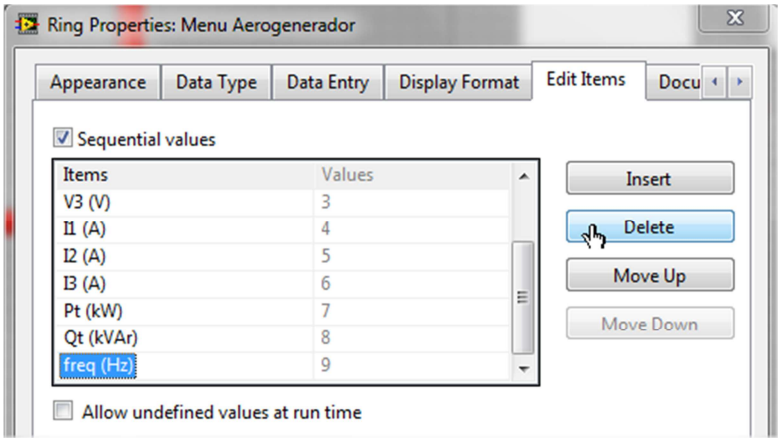

Figura 4.33: Eliminar el elemento freq del Menú Aerogenerador

El índice dentro del array de datos (DataPC) de nuestro módulo no habrá cambiado, porque sus datos seguirán comenzando en la misma posición. Sin embargo, excepto en la ocasión en la que el módulo a modificar sea el último en la posición de los datos, el índice de todos cuyos datos vengan después se habrá desplazado una posición.

Es decir, si Aerogenerador comienza en la posición 100 y contiene 10 datos, además del código del módulo, el siguiente comenzará en la posición 111. Sin embargo, si ahora Aerogenerador tiene 9 datos, el siguiente tendrá que comenzar en el 110, y así con todos los que estén después.

El valor del índice debe valer siempre 2 valores menos que la posición de la primera variable. Para comprobar el funcionamiento interno de la función creada MenuToChart consultar en la sección 3.2.10 la librería ChartToolsLLB.

Sin embargo, excepto la variación de los índices en los módulos, eliminar un atributo no produce ningún otro cambio en el Esquema General.

iii. Esquema General Térmico (EsquemaGeneralTermico.vi)

Dado que el diagrama de bloques de este esquema es idéntico al del Esquema General, a la hora de modificarlo basta con seguir los pasos definidos para éste.

iv. Esquema de Comunicación (EsquemaComunicacion.vi)

En el esquema de comunicación tenemos que variar la cantidad de datos que se insertarán en la relación de la base de datos correspondiente el módulo tratado, ya que ahora alojaremos un atributo menos. Es por eso que tenemos que modificar el código correspondiente a la Figura 4.34.

Figura 4.34: Código a modificar al añadir nuevo atributo

En este caso se trata del módulo Aerogenerador y su índice anterior valía 100 y su número de datos era 11 (código del módulo en la posición 100 y luego 10 datos). Si hemos eliminado un nuevo atributo, la posición del código no variará pero si lo hará la cantidad de datos. Es decir, ahora el número de datos será 10 en lugar de 11. Y antes la salida era por Data10, pero ahora, al ser el número de datos 10 la salida se dará por Data9.

Como se explicaba en el Esquema General de esta misma sección, habrá que cambiar los índices de los módulos cuyos datos se recojan después del módulo que hayamos modificado. A la hora de modificar los índices, seguir la norma general por la que el valor del índice debe valer siempre 2 valores menos que la posición de la primera variable.

Excepto los valores de los índices, no hay que realizar ningún otro cambio en el Esquema de Comunicación para que funcione correctamente con la nueva configuración de la Microrred.

v. Gráficas en Tiempo Real (RealTimeCharts.vi)

En la gráfica en tiempo real del módulo que estamos modificando debemos no poder mostrar el atributo que eliminamos. Para ello, primero debemos modificar el menú en el que se muestran las posibilidades como se ha explicado en el Esquema General de este mismo apartado.

Además, como en todos los demás programas, hay que modificar el índice de los módulos cuyos datos se envían después del modificado, ya que si no la función MenuToChart no devolverá los resultados correctos. Como norma general, el valor del índice debe valer siempre 2 valores menos que la posición de la primera variable. Para comprobar el funcionamiento interno de la función creada MenuToChart consultar en la sección 3.2.10 la librería ChartToolsLLB.

vi. Históricos (HistoricosMicrorred.vi)

Debemos dejar de mostrar la opción de recoger los datos del atributo que hemos eliminado.

Además, hay que modificar los argumentos de entrada de la función SelectedArray correspondiente al módulo del que se ha eliminado un atributo. Hay que eliminar el CheckBox, que corresponde al booleano con el nombre de la variable. Se eliminará su configuración asociada, si bien pueden quedar cables sin unir que se eliminan con la combinación de teclas Ctrl + B. También habrá que eliminar su dato correspondiente del array, haciendo clic derecho sobre él y seleccionando Data Operations / Delete Element. Como consecuencia, el array de entrada a la función SelectedArray tiene una posición menos, por lo que habrá que reducir la función BuildArray en una posición. Todos estos cambios se ven en la Figura 4.35.

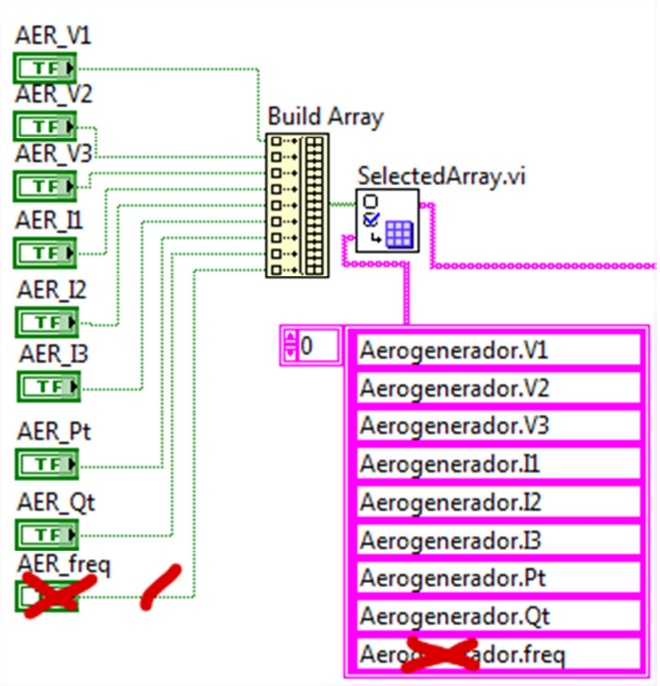

Figura 4.35:Cambios en las entradas de SelectedArray al eliminar un atributo de Aerogenerador

vii. Selección de Datos (SelectedData.vi)

Dado que el diagrama de bloques de este programa se basa en el de HistoricosMicrorred.vi, hay que realizar los mismos cambios que en éste.

viii. FTP (RecolectarFTP.vi)

Si se ha eliminado un atributo a un módulo ya existente se considera que los ficheros TDMS que se escriban con los datos han variado en tamaño, por lo que hay que realizar algún cambio.

Primero, en la función creada TDMStoDatabase (apartado 3.2.10, librería DBToolsLLB), en la entrada de la función ReshapeArray, que corresponde a la Figura 4.36.

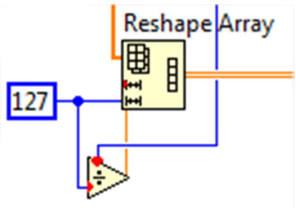

Figura 4.36: Entrada de Reshape Array

Hay que actualizar el indicador numérico, en este caso 127, al número total de datos, que será uno menor que el actual. Para comprobar su corrección se aconseja ejecutar el SubVI por separado con un fichero TDMS actualizado con los nuevos datos y comprobar el indicador OutPutArray, de modo que en la primera posición se mantenga el doble con la fecha y la hora, y el resto de datos también estén en su lugar.

El siguiente SubVI a modificar es InsertDB (apartado 3.2.10, librería DBToolsLLB). Realmente, este SubVI se ha creado a partir del Esquema de Comunicación, por lo que los cambios que se realicen serán los mismos que en éste.

4.4. Cambios en la Base de Datos

La base de datos ha sido diseñada e implementada de modo que admita gran cantidad de datos y trate los mismos eficientemente. Toda la información respecto al diseño de la Base de Datos se puede consultar en la sección 3.1.

Aunque la base de datos ha sido dotada con planes de mantenimiento (sección 3.1.3) que comprueban el correcto funcionamiento de la misma, entre otros, ocasionalmente se puede producir algún error grave que derive en la pérdida de datos, y por lo tanto, haya que restaurarla. Para que se ejecuten los planes de mantenimiento, SQL Server Agent debe estar activo. Es conveniente comprobar de vez en cuando que está activo, porque, aun cuando se haya preestablecido su activación automática, no conviene en ningún caso que no esté activado ya que se dejarían de hacer copias de seguridad, entre otros. Para comprobar que está activado, acceder al interfaz de SQL Server y conectarse con las opciones por defecto. Desplegando el árbol que aparece a la izquierda, en último lugar aparece SQL Server Agent. Deberá mostrarse como en la Figura 204.

Figura 204: SQL Server Agent activado

En caso de que en lugar de una flecha verde veamos una flecha hacia abajo roja, hacer clic derecho y seleccionar Start, aceptando en la ventana emergente que nos pide confirmar la operación.

Se definen en este manual los pasos a seguir para obtener información general de la base de datos, restaurarla y eliminarla.

Todos estos pasos serán llevados a cabo sobre el interfaz de SQL Server y hay que tener especial cuidado al manejarse en él, ya que ejecutar acciones sin antes comprenderlas completamente puede resultar fatal para la Base de Datos.

4.4.1. Información Base de Datos

Conviene revisar de vez en cuando la información sobre la base de datos para comprobar que los espacios ocupados resultan coherentes con lo esperado.

Se ha creado un script llamado InfoDB.sql en que contiene las siguientes instrucciones:

```
sp_helpdb Microrred

exec sp_spaceused
```

exec sp_spaceused Aerogenerador

El script se encuentra en la ruta:
C:\Users\USER\Desktop\RT Microrred\BASE_DATOS\Info\InfoDB\Scripts

Mediante estas instrucciones se puede ver información general de la base de datos.

La primera proporciona información general sobre el tamaño de la Base de datos y el archivo de log. Si el archivo de log llega a su límite no se podrán seguir produciendo inserciones. En principio, el plan de mantenimiento asegura que el archivo de log reduzca su tamaño, pero es conveniente hacer un seguimiento de su tamaño.

La segunda informa de tamaño utilizado por los datos, por los índices y la cantidad de espacio reservada.

Esta sentencia se puede ejecutar seguida del nombre de uno de los módulos, como se ve en la tercera, de modo que proporciona información concreta de esa relación. Nos informará del número de columnas existentes, el tamaño de los datos, de los índices y el espacio reservado.

4.4.2. Crear Base de Datos

Se han elaborado una serie de scripts para la creación de la base de datos y de sus relaciones. El script de creación de la base de datos se llama CreateDatabase y se encuentra en:
C:\Users\USER\Desktop\RT Microrred\BASE_DATOS\Info\InfoDB\Scripts

Si el interfaz de SQL Server se encuentra abierto, haciendo doble clic sobre el script se copiará el código en el interfaz de SQL Server de modo que solo haya que ejecutarlo, pulsando Execute, que se encuentra en la barra de herramientas, encima de la ventana de edición de consultas, como se ve en la Figura 4.37.

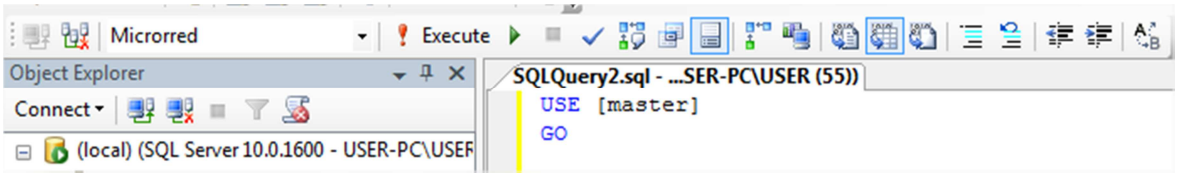

Figura 4.37: Barra de herramientas con la opción Execute.

Otro modo de ejecutar es usar el atajo de teclado F5. Una vez creada la base de datos, a la hora de ejecutar los demás scripts, asegurarse que la base de datos sobre la que se ejecutan es Microrred, como se ve en la Figura 207, ya que aparece en el desplegable al lado del botón Execute.

Después hay que ejecutar los scripts correspondientes al resto de relaciones.

Es posible ejecutar un único script llamado CreateTables y ubicado en: C:\Users\USER\Desktop\RT Microrred\BASE_DATOS\Info\InfoDB\Scripts

En caso de querer crear cada relación por separado, porque no se deseen crear todas, por ejemplo, se pueden ejecutar los scripts que se encuentran en : C:\Users\USER\Desktop\RT Microrred\BASE_DATOS\Info\InfoDB\Scripts\Tables

En este caso, ejecutar primero el script de la relación Momento(Momento) y a continuación el resto de scripts que se describen a continuación.

- Aerogenerador
- Baterias
- Cargas_Criticas
- Cargas_No_Criticas
- Fotovoltaica
- FRE_CAL_ACS
- Grupo_Diesel
- Hibrido_Cargador
- Hibrido_Eolica
- Hibrido_Inversor
- Hidrogeno
- Meteorologicas
- RED
- Salida_Inversor
- Ultracondensadores

4.4.3. Eliminar datos

Si se desea eliminar los datos de la base de datos se puede recurrir al script creado DeleteData. Contiene las siguientes instrucciones:

```
DELETE FROM Aerogenerador;
DELETE FROM Baterias;
DELETE FROM Cargas_Criticas;
DELETE FROM Cargas_No_Criticas;
DELETE FROM Fotovoltaica;
DELETE FROM FRE_CAL_ACS;
DELETE FROM Grupo_Diesel;
DELETE FROM Hibrido_Cargador;
DELETE FROM Hibrido_Eolica;
DELETE FROM Hibrido_Inversor;
DELETE FROM Hidrogeno;
```

```
DELETE FROM Meteorologicas;
DELETE FROM RED;
DELETE FROM Salida_Inversor;
DELETE FROM Ultracondensadores;
DELETE FROM Momento;
```

El script se encuentra en la ruta:
C:\Users\USER\Desktop\RT Microrred\BASE_DATOS\Info\InfoDB\Scripts

Los datos de la relación Momento deben ser eliminados en último lugar ya que las claves primarias de las demás relaciones hacen referencia a los datos de Momento. Si existen datos referenciados no nos dejará eliminarlos, por lo que hay que hacerlo en último lugar, cuando ningún dato dependa de ellos.

Tener mucho cuidado con la manipulación de este script y no ejecutarlo si no es estrictamente necesario, ya que hacerlo, obviamente, eliminaría todos los datos almacenados en la base de datos.

En caso de eliminación accidental se puede restaurar la base de datos consultando la información del apartado 4.3.5.

4.4.4. Eliminar Base de Datos

Para eliminar la base de datos, junto con todos sus datos asociados, acceder al interfaz de SQL Server con las opciones por defecto.

Expandir el árbol situado en la parte izquierda de la pantalla hasta llegar a Databases/Microrred.

Hacer clic derecho en Microrred y seleccionar Delete. La base de datos no es muy grande, pero la cantidad de datos almacenados es considerable, por lo que eliminarla es un proceso lento.

Si únicamente se desean eliminar las relaciones con todos sus datos asociados, pero no la base de datos (que estará vacía), ejecutar el script DropTables, ubicado en: C:\Users\USER\Desktop\RT Microrred\BASE_DATOS\Info\InfoDB\Scripts

4.4.5. Restaurar Base De Datos

La Base de Datos tiene una serie de Planes de Mantenimiento que por un lado comprueban el correcto funcionamiento de la misma, y por otro realizan copias de seguridad de los datos almacenados.

Estas copias de seguridad se realizan actualmente en el mismo disco duro en el que se almacenan los datos, por falta de un segundo sistema de almacenamiento. Sin embargo, sería altamente recomendable disponer de un segundo dispositivo de almacenamiento que guardase las copias de seguridad. Esto es así porque dentro de los riesgos de perder los datos, lo más probable es el fallo del sistema operativo o del disco duro, y, en ambos casos, se perderían tanto los datos de la base de datos como de las copias de seguridad.

Suponiendo que se instale un segundo disco duro y se redirijan las copias de seguridad ahí, tendríamos un sistema de copias altamente seguro frente a pérdidas. La copia de seguridad también nos permitiría recuperar los datos en caso de haber modificado la base de datos erróneamente y haberla corrompido, dejándola inutilizable. También se puede restaurar la base de datos en otro ordenador si se desea cambiar la ubicación de la misma.

Las copias de seguridad que se almacenan tienen una antigüedad máxima de 1 mes, para evitar que ocupen demasiado espacio. Es por eso que no se podrá restaurar la base de datos a un punto anterior a un mes de antigüedad.

Se producen tres tipos de copias de seguridad:

- Completa (Full): en una copia de seguridad completa se almacena tanto la estructura de la base de datos como todos los datos almacenados.
- Incremental (Differential): en una copia de seguridad incremental se almacenan únicamente los cambios en la estructura de la base de datos y los datos desde la última copia de seguridad completa o incremental.
- Archivo de log de transacciones (log transaction file): se almacenan las sentencias de transacción (inserciones, borrados, modificaciones,…) que se han ejecutado sobre la base de datos desde la última copia de seguridad, ya sea completa, diferencial o de archivo de log.

Al iniciar la restauración tenemos dos opciones: restaurar una base de datos existente o crear una nueva base de datos con los últimos datos. Si se ha producido algún error grave con la base de datos, para evitar futuras complicaciones, se recomienda eliminar la base de datos actual y realizar una restauración completa. Para eliminarla hacemos clic derecho en ella y elegimos Delete. La base de datos no es muy grande, pero la cantidad de datos almacenados es considerable, por lo que las acciones de borrado y restauración son lentas.

Para iniciar la restauración, conectarse al interfaz de SQL Server con las opciones por defecto, como se ve en la Figura 4.38.

Figura 4.38: Conexión al interfaz SQL Server

Los datos para la conexión son:

- Server type: Database Engine
- Server name: (local)
- Authentication: Windows Authentication
  - User name: USER-PC\USER
  - Password: *(no hay password)*

Expandir el árbol a la izquierda de la pantalla y hacer clic en Databases / Restore Database, como se ve en la Figura 4.39.

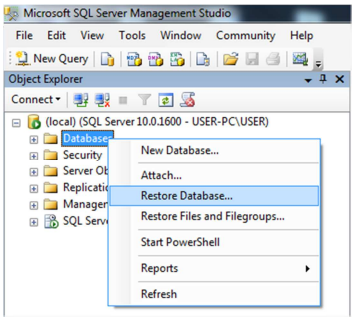

Figura 4.39: SQL Server, Restore Database

Aparecerá una ventana emergente con distintas opciones, como se puede ver en la Figura 4.40.

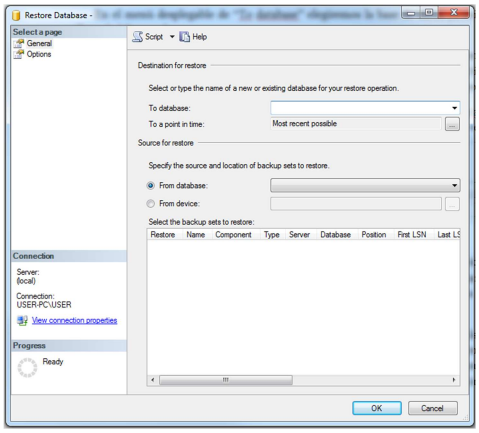

Figura 4.40: SQL Server, opciones Restore Database

a) Restaurar base de datos existente

Una de las opciones comentadas es la restauración de una base de datos existentes, porque, por ejemplo, se hayan borrado por error los datos. En este caso, en el apartado "Destination for restore" estableceremos las opciones de la Figura 4.41.

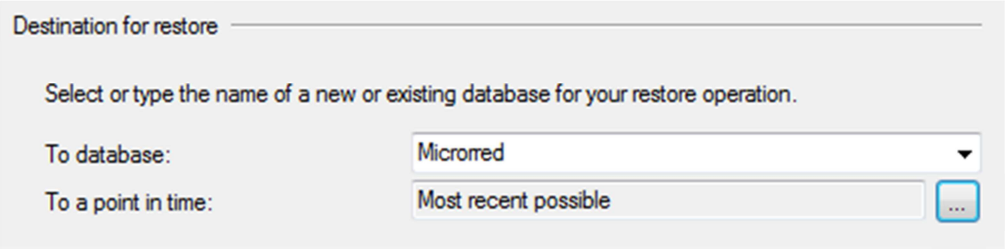

Figura 4.41: Restaurar Base de Datos existente

En el menú desplegable de "To database" elegiremos la base de datos ya existente llamada Microrred.

En el apartado "To a point in time" elegiremos "Most recent posibble". Si hacemos clic en el botón ubicado al final de esta opción podremos elegir un punto específico del tiempo, pero en principio se entiende que querremos recuperar la mayor cantidad de datos posibles.

En el apartado "Source for restore" seleccionamos la primera opción (From database), y, en el desplegable, el nombre de nuestra base de datos: Microrred, como se ve en la Figura 4.42.

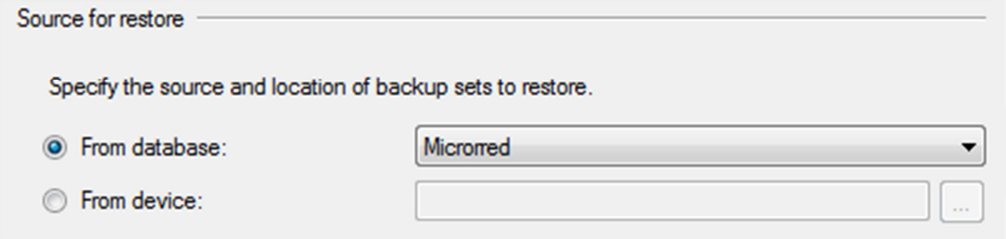

Figura 4.42: Restaurar desde base de datos existente

Esta acción seleccionará de la ubicación del disco duro de las copias de seguridad las correctas de acuerdo a nuestras opciones. Es posible que en el cuadro inferior aparezcan diversos archivos, correspondientes tanto a copias de seguridad completas como incrementales y del log de transacciones.

Con los archivos preseleccionados por SQL Server según nuestras opciones pulsar OK y esperar a la finalización del proceso. Al finalizar se puede comprobar la información general de la base de datos para comprobar que realmente se ha restaurado correctamente y con los datos oportunos. Consultar apartado 4.3.1 sobre cómo obtener información de la base de datos.

b) Restauración completa

En este tipo de restauración se parte de un SGBD sin base de datos, ya que, o bien ha quedado inutilizada y la hemos eliminado o se trata de una nueva ubicación de la base de datos, en otro ordenador, por ejemplo.

Dentro de las opciones generales, en el apartado “To database”, escribir el nombre de la nueva Base de Datos. El nombre debería ser Microrred ya que las conexiones ODBC se han realizado sobre la base de datos con ese nombre. Para más información sobre conexiones ODBC consultar sección 3.1.1.3. En caso de que únicamente hayamos perdido datos y queramos recuperarlos en la base de datos que ya existe, nombrada como Microrred, seleccionar del desplegable

En la sección “Source for restore” seleccionar la opción “From device” y hace clic como se ve en la Figura 4.43.

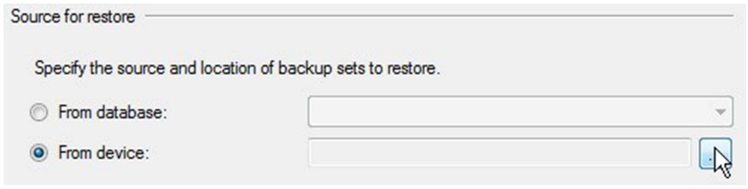

Figura 4.43: Restauración desde dispositivo

Nos aparece una ventana para seleccionar los archivos a restaurar. Elegir opción desde fichero y añadir (Add), como se ve en la Figura 4.44.

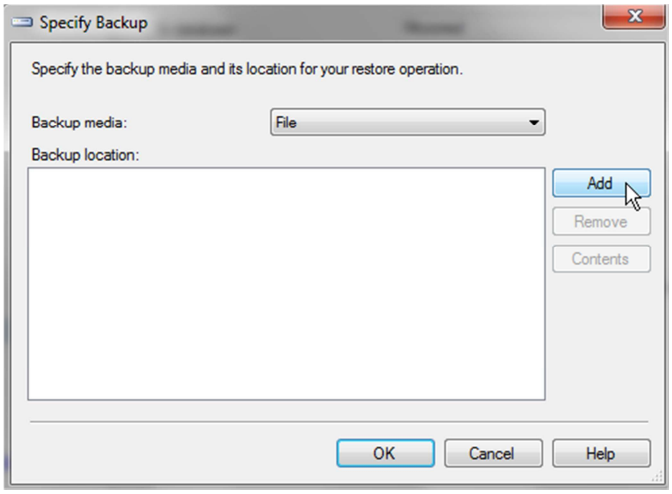

Figura 4.44: Restaurar desde fichero

Buscar en el sistema de ficheros la carpeta con las copias de seguridad. Actualmente se trata de C:\Program Files\Microsoft SQL Server\ MSSQL10.MSSQLSERVER \MSSQL\ Backup. Podemos acceder directamente desde el explorador de Windows a esa ubicación de modo que podamos comprobar los detalles de los archivos.

Debemos localizar la última copia completa (Full BackUp), la última incremental (Differential BackUp) y, desde ella, todas las copias del log de transacciones (Log Transaction BackUp) hasta la fecha actual.

Las copias de seguridad completas se realizan 1 vez por semana, el Domingo a las 00:30h.

Las copias de seguridad incrementales se realizan todos los días a las 00:15h.

Las copias de seguridad del log de transacciones se realizan todos los días cada 8h (3 por día).

Así, según el día en el que nos encontremos, podemos deducir la fecha de realización de las copias de seguridad que necesitamos, y mostrando los detalles de los archivos descubrir cuáles son los interesantes.

En los siguientes pasos se habla de opción Recovery y No Recovery. Para cambiar entre ellas, una vez seleccionado el archivo a restaurar, ir a Opciones (Options) en la parte superior izquierda de la ventana y seleccionar Recovery o No Recovery, según lo deseado, como se muestra en la Figura 4.45.

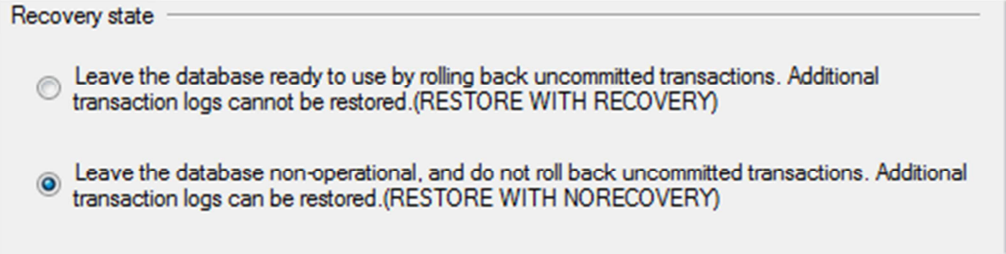

Figura 4.45: Selección de Recovery o No Recovery

La diferencia entre ellas es que la primera no permite realizar más restauraciones del log de transacciones, por lo tanto, sólo lo realizaremos con el último de los archivos.

Ejecutar cada restauración por separado siguiendo el siguiente esquema:

i. Seleccionar la última copia de seguridad completa (Full) en la pantalla especificada en la Figura 208. Pulsar en OK y aparecerá el fichero en el recuadro de la ventana de restauración, titulado “Select the bakcup sets to restore”. En las opciones comentadas en la Figura 209 seleccionar “RESTORE WITH NORECOVERY”. Pulsar en OK y esperar a que finalice el proceso.
ii. Seguir el mismo proceso con la última copia de seguridad incremental (Differential). También debe tener en Opciones “RESTORE WITH NORECOVERY”. Pulsar en OK y esperar a que finalice el proceso.
iii. Restaurar por separado y exceptuando la última cada una de las copias del fichero de log (Log Transaction) con la opción “RESTORE WITH NORECOVERY”. Pulsar en OK y esperar a que finalice el proceso.
iv. Restaurar la última copia del fichero de log (Log Transaction) con la opción “RESTORE WITH RECOVERY”. También en la pantalla de opciones, elegir reemplazar actual (Figura 4.46). Pulsar en OK y esperar a que finalice el proceso.

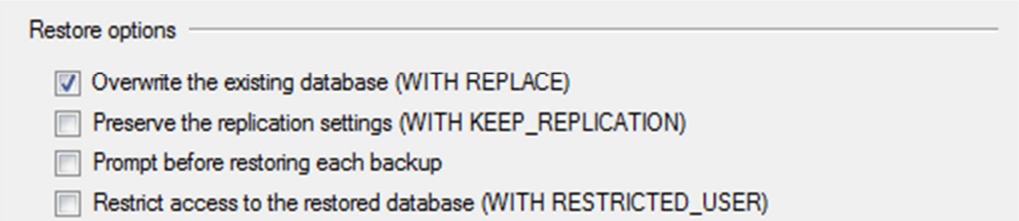

Figura 4.46: Reemplazar base de datos existente

A tener en cuenta que si sólo hay una copia del log de transacciones, ésta deberá restaurarse con la opción “Recovery”. Si no disponemos de ninguna copia de seguridad del log de transacciones, será la copia de seguridad incremental (Differential) la que tengamos que restaurar con la opción “Recovery”. En caso de no disponer tampoco de copia de seguridad diferencial, la copia de seguridad completa (Full) será el único fichero a restaurar, y se hará con la opción “Recovery”.

En resumen, el último fichero a restaurar deberá llevar la opción “Recovery”, mientras que los demás llevarán la opción “No Recovery”.